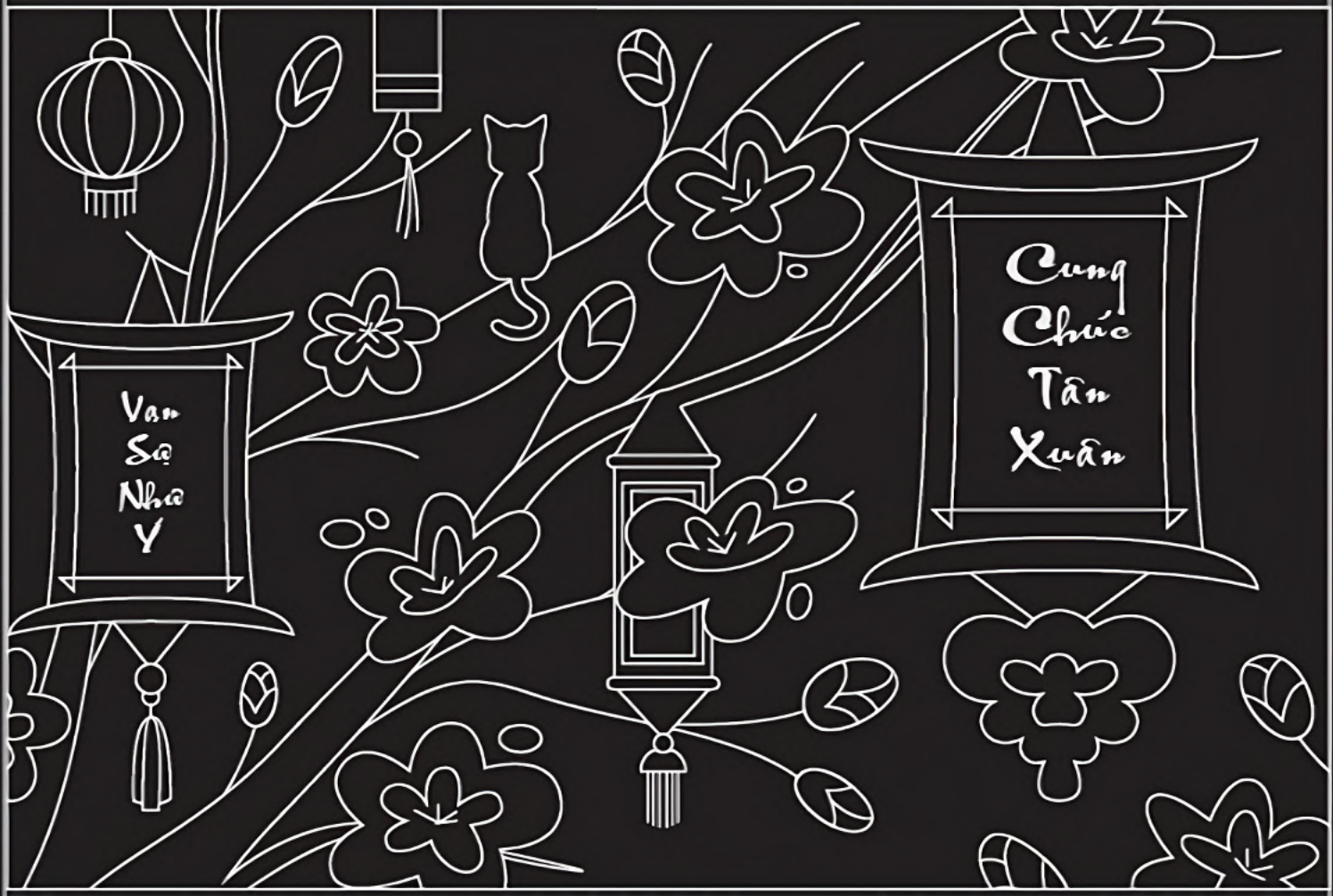
Tạp chí

VNOI

Xuân Quý Mão



Số Đặc Biệt

Quý độc giả thân mến,

Tạp chí VNOI - Xuân Quý Mão là một ấn phẩm đặc biệt của Câu lạc bộ Olympic Tin học Việt Nam, như một món quà đầu xuân mà VNOI muốn dành tặng cho cộng đồng nhân dịp Tết Quý Mão 2023.

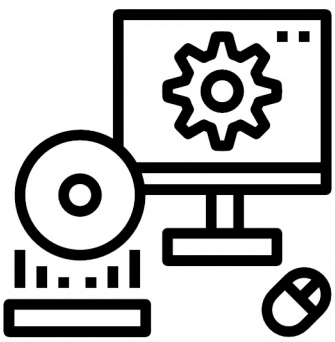
Với mục đích truyền tải kiến thức, kinh nghiệm qua nhiều chủ đề, số tạp chí đặc biệt kì này sẽ mang đến cho các bạn những bài viết thú vị với nội dung đa dạng, thân thiện và phù hợp với nhiều đối tượng độc giả. Tạp chí bao gồm 11 bài viết học thuật xoay quanh các chủ đề trong Tin học và Lập trình thi đấu đến từ 10 tác giả, được chúng mình tuyển chọn qua rất nhiều bài viết được gửi về từ khắp mọi miền tổ quốc. Bên cạnh những nội dung mang tính học thuật, tạp chí kì này còn có sự xuất hiện của 6 vị khách mời đặc biệt qua 6 bài phỏng vấn, hứa hẹn mang tới cho các bạn độc giả những câu chuyện ý nghĩa về hành trình học Tin của các nhân vật có tầm ảnh hưởng lớn trong cộng đồng Tin học Việt Nam.

Trong niềm vui hân hoan, nao nức của xuân mới, ban biên tập nói riêng và tập thể VNOI nói chung xin chân thành gửi lời tri ân đến các khách mời, các tác giả, quý độc giả đã luôn dành quan tâm, theo dõi và ủng hộ trong suốt thời gian qua.

Nhân dịp xuân Quý Mão 2023, tập thể VNOI thân chúc bạn đọc một năm mới sức khỏe, hạnh phúc và nhiều thành công!

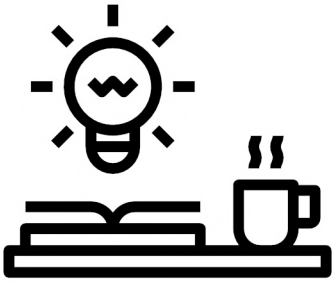
BAN BIÊN TẬP

Mục lục



Làm sao để "chơi game"?

Trần Xuân Bách
12A1 Tin, Trường THPT Chuyên Khoa học Tự nhiên, Đại học Quốc gia Hà Nội



Giới thiệu

Lí thuyết trò chơi (game theory) là một công cụ được dùng để nghiên cứu các tình huống xã hội (cuộc chơi) giữa các bên cạnh tranh nhau. Trên một vài khía cạnh, lí thuyết trò chơi là khoa học về chiến lược, về những tình huống chiến thuật trong đó người chơi chọn các hành động khác nhau để cố gắng thay đổi kết quả theo hướng có lợi cho mình, hay nói cách khác, tối ưu hóa kết quả.

Một trong những bài toán nổi tiếng nhất là Song đề tù nhân (Prisoner's dilemma) [1] . A và B bị bắt vào tù, và mỗi người có hai lựa chọn - đầu thú hay không đầu thú - với những kết cục khác nhau. Nhìn thoáng qua thì trò chơi trông rất đơn giản, tuy nhiên ẩn trong đó là hai hướng suy nghĩ tuy đều hợp lí, song lại đối lập nhau.

Tội phạm B \ Tội phạm A	Thú nhận	Không nhận
Thú nhận	20 năm tù – 20 năm tù	Tha bổng – Tử hình
Không nhận	Tử hình - Tha bổng	5 năm tù – 5 năm tù

Ví dụ của Song đề tù nhân

Trò chơi và lí thuyết trò chơi cũng rất phổ biến trong lập trình thi đấu vì sự đa dạng về độ khó và thể thức của nó. Tuy nhiên, lí thuyết trò chơi [2] chỉ đề cập đến những dạng *trò chơi tổ hợp cân bằng*, mà không bao phủ những dạng khác có thể xuất hiện.

Trong bài viết này, ta sẽ xem xét qua một số trò chơi sử dụng tính chất **đối xứng**. Khi "môi trường" của trò chơi là một bảng một đến hai chiều nào đó, thì một chiến thuật thường xuất hiện sẽ là chơi đối xứng qua tâm hoặc trục đối xứng của bảng. Ngoài ra, đôi lúc ta sẽ sử dụng việc một vài tính chất có **vai trò tương tự nhau**.

CF 630R Game

Đề bài

Hai người A và B chơi một trò chơi như sau:

Cho một bảng hình vuông có kích thước $n \times n$. Ở một lượt, người chơi tô màu một ô vuông mà **không** chung cạnh với bất kì ô vuông nào đã được tô màu từ trước. Ai không tô được ô vuông nữa là người thua cuộc.

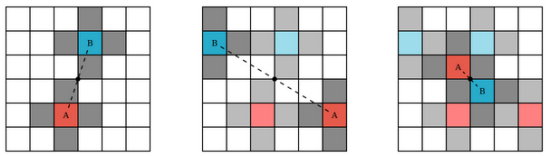
Hai người lần lượt chơi với nhau, với A là người đi đầu tiên. Biết rằng cả hai người chơi tối ưu, hỏi ai là người thắng cuộc?

Giới hạn: $1 \leq n \leq 10^{18}$.

Lời giải

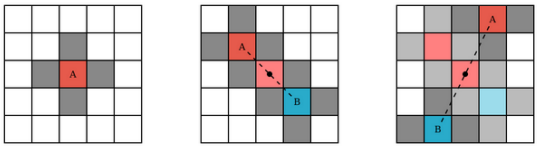
Đánh số các ô trong bảng từ $(1, 1)$ đến (n, n). Xét tính chẵn lẻ của n:

- Khi n chẵn, B sẽ thắng.
  Nếu A tô màu ô (x, y), thì B chỉ cần tô màu ô $(n + 1 - x, n + 1 - y)$ - ô đối xứng với (x, y) qua tâm bảng.
  Vì hai ô đối xứng nhau không bao giờ kề cạnh nhau, và trạng thái của bảng luôn đối xứng qua tâm (nếu A tô màu được thì B cũng tô màu được), nên B luôn luôn đi được như vậy. Vậy A phải là người không tô được đầu tiên, tức là B là người thắng cuộc.



Minh họa chiến thuật của B

- Khi n lẻ, A sẽ thắng.
  Ở lượt đầu tiên, A sẽ tô màu ô $\left(\left\lceil \frac{n}{2} \right\rceil, \left\lceil \frac{n}{2} \right\rceil\right)$ - ô ở tâm của bảng. Sau đó, A chỉ việc đi đối xứng với B qua tâm. Tương tự như lập luận khi n chẵn, ta có thể thấy A là người thắng cuộc.



Minh họa chiến thuật của A

CF 197A Plate Game

Đề bài

Hai người A và B chơi một trò chơi như sau:

Cho một tờ giấy hình chữ nhật kích thước n (cm) $\times$ m (cm). Trong một lượt, người chơi cắt đi một hình tròn bán kính l (cm) nguyên vẹn, không được khuyết ở chỗ nào. Ai không cắt được hình tròn nữa là người thua cuộc.

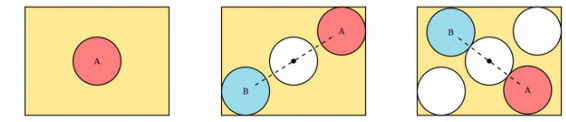
Hai người lần lượt chơi với nhau, với A là người đi đầu tiên. Biết rằng cả hai người chơi tối ưu, hỏi ai là người thắng cuộc?

Giới hạn: $1 \leq n, m, l \leq 100$.

[1] https://en.wikipedia.org/wiki/Prisoner%27s_dilemma
[2] https://vnoi.info/wiki/algo/math/game-theory.md

Lời giải

Cũng tương tự như bài trên, ta có thể đoán chiến thuật tối ưu của A là đặt một hình tròn vào tâm của tờ giấy, rồi đi đối xứng với B trong các nước đi còn lại.



Minh họa chiến thuật của A

Tuy nhiên, đừng vội vàng mà nghĩ rằng A sẽ thắng trong mọi trường hợp! Nếu như $n < 2l$ hoặc $m < 2l$, thì ngay từ đầu A đã không thể cắt một hình tròn nguyên vẹn ra khỏi tờ giấy.

CF 1451D Circle Game

Đề bài

Hai người A và B chơi một trò chơi như sau:

Có một đồng xu lúc đầu ở điểm $(0, 0)$ trên mặt phẳng hai chiều. Ở một lượt, giả sử đồng xu đang ở tọa độ (x, y), thì người chơi được phép di chuyển đồng xu bằng một trong hai cách như sau:

- $(x, y) \to (x + k, y)$
- $(x, y) \to (x, y + k)$

Sau khi di chuyển, đồng xu phải nằm trong hình tròn tâm $(0, 0)$ bán kính d. Nói cách khác, trong mọi thời điểm, $x^2 + y^2 \le d^2$. Ai không di chuyển được đồng xu nữa là người thua cuộc.

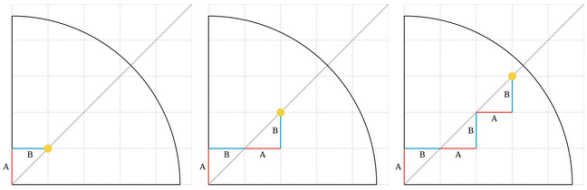
Hai người lần lượt chơi với nhau, với A là người đi đầu tiên. Biết rằng cả hai người chơi tối ưu, hỏi ai là người thắng cuộc?

Giới hạn: $1 \le k \le d \le 10^5$.

Lời giải

Do trong một lượt ta chỉ được tăng tọa độ x hoặc tọa độ y của đồng xu, nên chắc chắn ta sẽ không thể chơi đối xứng qua tâm mặt phẳng (điểm $(0, 0)$) được rồi.

Tuy nhiên, ta có thể thấy rằng trục x và y có vai trò như nhau. Do vậy, nếu như một người chơi tăng x lên k ở lượt của mình, thì người còn lại có thể tăng y lên k ở lượt tiếp theo. Nói cách khác, người chơi có thể giữ đồng xu ở trên một đường chéo nào đó.

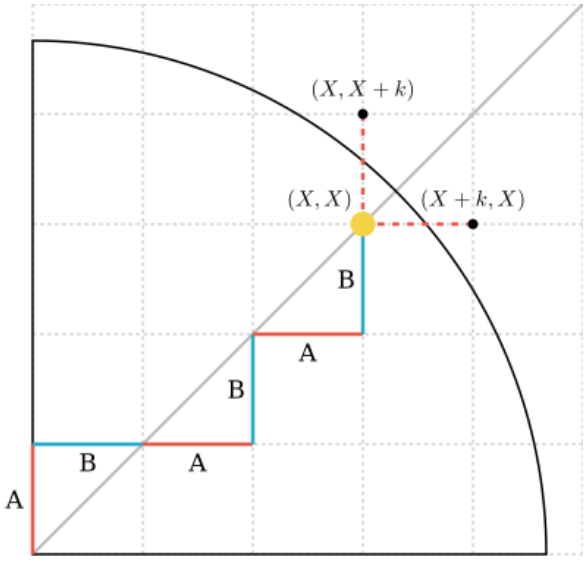


Minh họa chiến thuật giữ trên đường chéo

Phát triển ý tưởng này thêm sẽ đưa ta đến lời giải:

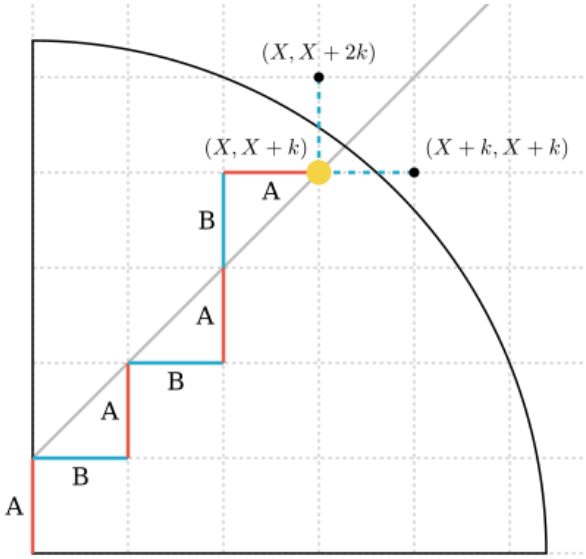
Gọi X là giá trị cao nhất thỏa mãn X chia hết cho k và điểm (X, X) nằm trong hình tròn bán kính d. Khi đó:

- Nếu điểm $(X, X + k)$ nằm ngoài hình tròn bán kính d, thì B thắng.
  B sẽ làm chiến thuật trên để "giữ" đồng xu ở trên đường chéo $y = x$. Khi B di chuyển đồng xu đến điểm (X, X), thì A không còn nước đi hợp lệ nữa (do cả hai điểm $(X+k, X)$ và $(X, X+k)$ đều nằm ngoài hình tròn).

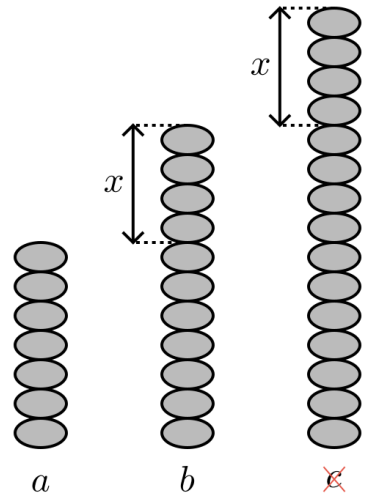


Minh họa chiến thuật của B

- Nếu điểm $(X, X + k)$ nằm trong hình tròn bán kính d, thì A thắng.
  Ở nước đi đầu tiên, A sẽ di chuyển đồng xu đến điểm $(0, k)$, sau đó "giữ" đồng xu ở trên đường chéo $y = x + k$. Khi đến điểm $(X, X + k)$, thì B chỉ còn hai nước đi có thể là $(X + k, X + k)$ và $(X, X + 2k)$.
  Tuy nhiên, theo định nghĩa của X, điểm $(X + k, X + k)$ nằm ngoài hình tròn. Ngoài ra, $(X^2 + (X + 2k)^2) - ((X + k)^2 + (X + k)^2) = 2k^2 > 0$, nên điểm $(X, X+2k)$ cũng nằm ngoài hình tròn tâm $(0, 0)$ bán kính d.



Minh họa chiến thuật của A



Để ý điều kiện chồng c đã bị chọn ở lượt trước

CF 1375F Integer Game

Đề bài

Hai người A và B chơi một trò chơi như sau:

Có ba chồng đá, lúc đầu mỗi chồng có lần lượt a, b, c viên đá, với a, b, c là ba số nguyên dương **đôi một phân biệt**. Ở mỗi lượt:

- A chọn một số nguyên dương x và đưa cho B số đó.
- B sẽ thêm x viên đá vào một trong ba chồng đá, với điều kiện là **B không được chọn cùng một chồng đá trong hai lượt liên tiếp**.

B sẽ thua nếu như, tại bất kì thời điểm nào, tồn tại hai chồng đá có cùng số lượng đá. A sẽ thua nếu như B chưa thua sau 1000 lượt chơi.

Biết rằng cả hai người chơi tối ưu, hỏi ai là người thắng cuộc?

Lời giải

Không mất tính tổng quát, giả sử $a < b < c$.

Đầu tiên, B ~~chơi đá~~ thua khi nào? Đó là khi ở lượt cuối cùng, ta có:

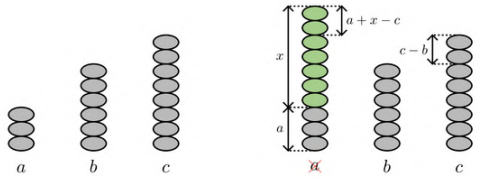
- $a + x = b$, nên B không được chọn chồng a. (1)
- $b + x = c$, nên B không được chọn chồng b. (2)
- Ở lượt trước đó, B đã thêm vào chồng c hiện tại, nên B không thể chọn lại chồng c nữa. (3)

Nhận thấy ở tình huống trên, a, b, c tạo thành một *cấp số cộng*. Nói cách khác, a và c đối xứng nhau qua b: $c - b = b - a = x$.

Vậy ở lượt trước đó, A sẽ đi như thế nào để "dồn" B vào trường hợp trên?

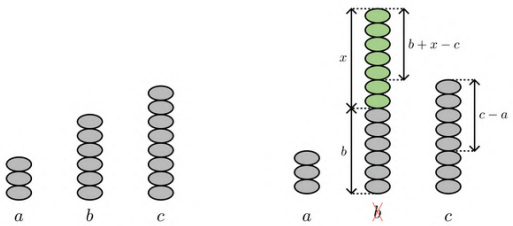
Giả sử ở lượt này Việt thêm vào chồng a một lượng là x, vậy số đá mới sẽ là $b < c < a + x$. Lưu ý rằng ta phải chọn x đủ lớn để thỏa mãn điều kiện 3. Để thỏa mãn điều kiện (1) và (2), ta có:

$$\begin{aligned} & a + x - \quad c \quad = c - b \\ \iff & \qquad\quad x \quad = 2c - a - b \end{aligned}$$



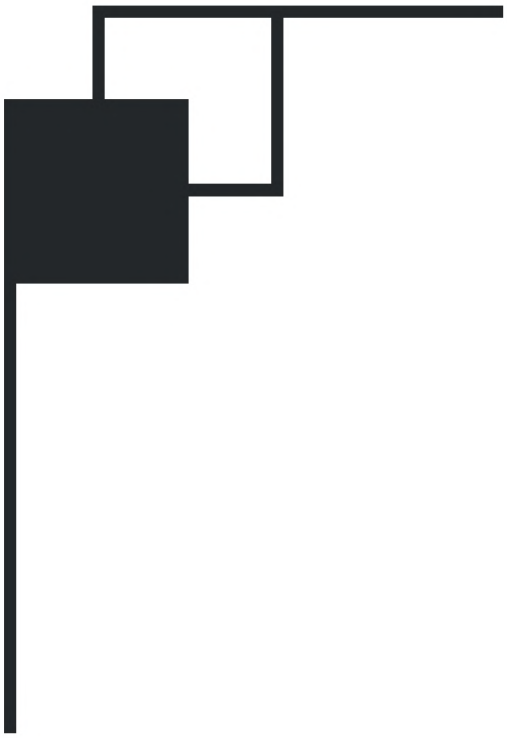
Để ý rằng ở phương trình trên, a và b **có vai trò tương tự nhau**. Nói cách khác, kể cả khi ở lượt này B thêm vào chồng b một lượng là $x = 2c - b - a$, thì giá trị mới của các chồng đá vẫn tạo thành một cấp số cộng.

Thật vậy, số đá mới của ba chồng đá sẽ là $a < c < 2c - a$, và $2c - a - c = c - a$.



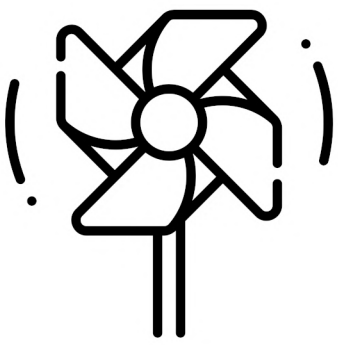
Như vậy, B chỉ còn lại một nước đi duy nhất là chọn chồng đá lớn nhất c. Tuy nhiên, nếu ta làm chiến thuật trên một lần nữa, thì theo luật B phải chọn một trong hai chồng còn lại.

Kết luận: Nếu cả hai người chơi tối ưu thì A có chiến thuật luôn luôn thắng B trong duy nhất ba lượt chơi.



Tất tần tật về Hash

Lê Tuấn Hoàng
12 Tin, Trường THPT Chuyên Hà Nội - Amsterdam



Giới thiệu

Hash (băm), là việc biến đầu vào có kích thước bất kì thành giá trị đầu ra có kích thước cố định.

Hash có tính ứng dụng rất lớn ở nhiều lĩnh vực trong thực tế. Trong lập trình thi đấu, ngoài ứng dụng việc kiểm tra tính bằng nhau các đối tượng có tính thứ tự như xâu, Hash cũng có thể dùng để kiểm tra tính bằng nhau của các tập hợp (không có tính thứ tự).

Trong rất nhiều bài toán, Hash có tốc độ thực thi nhanh hơn, cài đặt dễ hơn so với các thuật toán tất định.

Bài viết sẽ trình bày một số cách Hash tập hợp phổ biến cùng một số vấn đề liên quan.

Một chút về sinh số ngẫu nhiên

Các phương pháp trình bày ở dưới đều liên quan đến sinh số ngẫu nhiên, chất lượng số ngẫu nhiên có thể ảnh hưởng đến tính chính xác của thuật toán.

Một trong những phương pháp sinh số ngẫu nhiên phổ biến trên máy tính là sử dụng các công thức toán học để tạo ra một dãy số (pseudorandom number generator) từ với một seed cho trước (thường được lấy là thời gian của hệ thống). Seed giống nhau sẽ tạo ra các dãy giống nhau. Ví dụ ta có hàm $f(x) = x \times 3 \mod 11$, nếu chọn seed là 1, số đầu tiên của dãy là $f(1) = 3$, số thứ hai là $f(f(1)) = 9$, số thứ ba là $f(f(f(1))) = 5$...

Nếu không cẩn thận, các số được sinh ra sẽ mất tính ngẫu nhiên, nên đây là một vấn đề rất đáng lưu tâm.

Không chỉ trong phạm vi bài toán sử dụng phương pháp Hash, phần này áp dụng cho tất cả các bài toán có liên quan đến sinh số ngẫu nhiên.

Hàm `rand()`:

Khi mới làm quen với việc sinh số ngẫu nhiên trong C++, có lẽ hầu hết chúng ta đều sử dụng hàm `rand()`. Tuy nhiên hàm này chỉ có thể sinh ra số ngẫu nhiên trong đoạn $[0, RAND\_MAX]$ với `RAND_MAX` là một hằng số, tùy thuộc vào trình biên dịch, được đảm bảo có giá trị ít nhất 32767. Đây là một giá trị quá nhỏ, khả năng cao sẽ gây ra các kết quả sai. Chưa kể hàm `rand()` được cài đặt bằng một thuật toán tương đối đơn giản [3] .

Có thể sinh một số mới từ các giá trị sinh ra từ hàm `rand()`. Tuy nhiên cần lưu ý tính ngẫu nhiên của số mới tạo ra có thể không được cao. Ví dụ, nhân 2 giá trị sinh ra từ hàm `rand()` sẽ tạo ra số mới có 75% khả khả năng là số chẵn.

MT19937

Sử dụng thuật toán Mersenne Twister [4] , phát triển dựa trên số nguyên tố Mersenne $2^{19937} - 1$, cũng là chu kì của nó. Cài đặt của thuật toán trong C++ được biết với tên `mt19937`, có thể tạo ra số ngẫu nhiên tốt hơn hàm `rand()`.

Lưu ý là `mt19937` chỉ có thể sinh số nguyên 32-bit, nếu muốn sinh số nguyên 64-bit thì cần dùng `mt19937_64`.

Cách sử dụng:

```
#include <bits/stdc++.h>
using namespace std;

int main() {
  long long seed = time(0);
  mt19937 rng32(seed);
  cout << "random 32: " << rng32() << endl;
  mt19937_64 rng64(seed);
  cout << "random 64: " << rng64() << endl;
}
```

Ở đây `time(0)` chính là seed, là số giây tính từ 01/01/1970. Nên nếu chạy code trong cùng một giây thì sẽ được kết quả giống nhau. Trong các kì thi như HSGQG thì seed này là đủ tốt.

Để tránh bị hack khi tham gia contest trên các nền tảng như Codeforces[5] (bị các thí sinh khác biết được seed, tính trước được các giá trị sinh ra và sinh test chặn), có thể sử dụng seed là

```
const auto seed = chrono::steady_clock::now()
                    .time_since_epoch()
                    .count();
```

, cũng là thời gian nhưng có độ chính xác cao hơn, ít nhất đến mili giây, rất khó để có thể xác định được seed.

Các phương pháp Hash

Hash với phép XOR

Bài toán

Một dãy số $b_1, b_2, ..., b_k$ được gọi là hoàn hảo nếu nó là một hoán vị của $1, 2, ..., k$.

[3] https://en.wikipedia.org/wiki/Linear_congruential_generator

[4] https://en.wikipedia.org/wiki/Mersenne_Twister

[5] Codeforces: một trang web tổ chức các cuộc thi lập trình thi đấu với các dạng đề, bài tập đa dạng.

Cho một dãy số nguyên a độ dài $n \leq 10^5$ với $1 \leq a_i \leq n$ và $q \leq 10^5$ truy vấn có dạng (l, r), hãy cho biết dãy con $a_l, a_{l+1}, ..., a_r$ có phải là dãy số hoàn hảo hay không?

Nhận xét

Ta cần so sánh $a_l, a_{l+1}, ..., a_r$ và $1, 2, ..., r-l+1$, tính thứ tự không quan trọng.

Giải pháp

Ta sẽ gán một số nguyên khác không ngẫu nhiên $h(i)$ cho mỗi giá trị i từ 1 đến n.

Phép XOR có tính chất giao hoán, dễ thấy dãy con $a_l, a_{l+1}, ..., a_r$ hoàn hảo thì biểu thức sau phải thỏa mãn, tuy nhiên phải lưu ý rằng chiều ngược lại có thể không đúng:

$$h(1) \oplus h(2) \oplus ... \oplus h(r-l+1) = h(a_l) \oplus h(a_{l+1}) \oplus ... \oplus h(a_r)$$

Đây là bài toán cơ bản có thể dễ dàng xử lí bằng mảng tiền tố.

Có thể có phần tử lặp lại trong dãy a, nên nếu một phần tử xuất hiện chẵn lần sẽ giống với việc không xuất hiện lần nào. Tuy vậy nếu có phần tử xuất hiện nhiều lần thì mã Hash của dãy con sẽ khác với mã Hash của $h(1) \oplus h(2) \oplus ... \oplus h(r-l+1)$.

Khả năng xung đột

Mỗi bit trong phép XOR là độc lập. Xét riêng một bit, nếu 2 bit đầu vào được đảm bảo là ngẫu nhiên thì bit đầu ra sẽ có 50% là bit 0 và 50% là bit 1, khả năng xung đột sẽ là $\frac{1}{2}$. Vậy với k bit, khả năng xung đột sẽ là $\frac{1}{2^k}$ mỗi truy vấn.

Có thể dễ dàng sinh các số ngẫu nhiên 64-bit, khả năng xung đột mỗi truy vấn chỉ là $\frac{1}{2^{64}}$, đủ tốt cho các bài toán trong lập trình thi đấu.

Hash với phép cộng

Bài toán

Cho hai dãy số nguyên a độ dài n và b độ dài m ($n, m \leq 10^5$) với $1 \leq a_i \leq n$ và $q \leq 10^5$ truy vấn có dạng:

$$l\ r\ ku_1\ v_1 u_2\ v_2 ... u_k\ v_k$$

tổng k trong tất cả các truy vấn không quá 10^5.

Hãy cho biết dãy con $a_l, a_{l+1}, ..., a_r$ có gồm đúng v_1 số u_1, v_2 số u_2,...,u_k số v_k hay không?

Nhận xét

Giống với bài toán ở phần trước, ta cũng cần kiểm tra tính bằng nhau của hai tập hợp, thứ tự của các phần tử không quan trọng. Nhưng không thể dùng phép XOR vì phép XOR sẽ không bảo toàn được tần suất xuất hiện của các phần tử.

Giải pháp

Ta sẽ gán một số nguyên khác không ngẫu nhiên $h(i)$ cho mỗi giá trị i từ 1 đến n.

Thay vì dùng phép XOR, ta dùng phép cộng, dãy con $a_l, a_{l+1}, ..., a_r$ hoàn hảo khi, cũng giống phương pháp XOR, chiều ngược lại cũng có thể không đúng:

$$h(a_l) + h(a_{l+1}) + ... + h(a_r) = h(u_1) \times v_1 + h(u_2) \times v_2 + ... + h(u_k) \times v_k$$

Khả năng xung đột

Trong cài đặt thực tế, ta thường sử dụng một module M để tránh tràn số, hoặc thậm chí để tràn số luôn cũng không sao (khi đó $M = 2^{64}$ nếu ta để dữ liệu kiểu số nguyên 64-bit không dấu).

Nếu dụng module P là số nguyên tố lớn. Xét 2 multiset A và B khác nhau mà $h(A) \equiv h(B)$ và $C = A \cap B$ nên $h(A - C) \equiv h(B - C)$.

Giả sử x là số chỉ xuất hiện trong $A - C$ và y là số lần xuất hiện của x. Do $0 < y < P$ nên có thể sử dụng nghịch đảo modulo, ta có $h(x) \equiv \frac{1}{y}[h(B - C) - h(A - C - \{x\} \times y)]$. Vì $h(x)$ thuộc khoảng $[1, P)$ nên khả năng xung đột là $\frac{1}{P-1}$.

Tuy vậy kể cả khi module M không nguyên tố, khả năng xung đột cũng vẫn là khoảng $\frac{1}{M}$. Với M đủ lớn, hoặc khi thực hiện các phép tính tràn số ($M = 2^{64}$), khả năng này là đủ nhỏ.

[6] https://www.facebook.com/codingcompetitions/hacker-cup/2022/round-2/problems/A2

Ứng dụng trong một số bài toán

Bài toán 1 - Facebook hackerup 2022 Round 2 - A2 - Perfectly Balanced [6]

Trước khi đến với phần lời giải, bạn đọc có thể thử sức với bài toán.

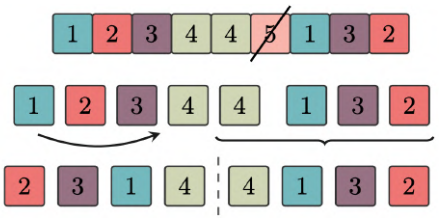
Tóm tắt đề bài

Một dãy số $b_1, b_2, ..., b_k$ gọi là hoàn hảo nếu k là số nguyên dương chẵn và có thể sắp xếp lại $b_1, b_2..., b_{\frac{k}{2}}$ sao cho b trở thành dãy đối xứng.

Cho một dãy A có độ dài N gồm các phần tử $1 \leq A_i \leq 10^6$ và Q truy vấn có một trong hai dạng sau:

- $1\ X_i\ Y_i$: Gán số nguyên $1 \leq Y_i \leq 10^6$ cho A_{X_i}.
- $2\ L_i\ R_i$: kiểm tra xem dãy con $A_{L_i...R_i}$ có tạo ra dãy "gần hoàn hảo" không?

Định nghĩa của dãy "gần hoàn hảo" là có thể xóa đi đúng 1 số của dãy đó để tạo ra được dãy hoàn hảo như ví dụ dưới.



Lời giải

Sử dụng phương pháp Hash với phép cộng. Ta có tập $H = \{h(1), h(2), ..., h(10^6)\}$, $f(l, r) = h(a_l) + h(a_{l+1}) + ... + h(a_r)$.

Dĩ nhiên ta chỉ xét dãy con có độ dài lẻ. Khi ấy với $m = \lfloor \frac{(l+r)}{2} \rfloor$ nếu $f(l, m) - f(m+1, r) \in H$ hoặc $f(m, r) - f(l, m-1) \in H$ thì dãy con "gần hoàn hảo", vì hiệu của hai phần chính là mã Hash của phần tử cần bị xóa.

Truy vấn 1 được xử lí bằng các Cấu trúc dữ liệu (CTDL), ví dụ Segment Tree [7] .

```cpp
#include <bits/stdc++.h>
using namespace std;

typedef unsigned long long ull;

const int maxn = 1e6 + 5;

int n;
ull rnd[maxn];
mt19937_64 rng(time(0));
unordered_set<ull> H;

// Segment Tree thực hiện thao tác cập nhật điểm
// và truy vấn đoạn
struct SegmentTree {
  vector<ull> node;
  SegmentTree(int n) : node(4 * n + 12) {}

  void update(int v, int l, int r, int pos,
              ull val) {
    if (r < pos || l > pos) return;
    if (l == r) {
      node[v] = val;
      return;
    }

    int m = (l + r) >> 1;
    update(v << 1, l, m, pos, val);
    update(v << 1 | 1, m + 1, r, pos, val);
    node[v] = node[v << 1] + node[v << 1 | 1];
  }

  ull get(int v, int l, int r, int tl, int tr) {
    if (r < tl || l > tr) return 0;
    if (tl <= l && r <= tr) return node[v];

    int m = (l + r) >> 1;
    return get(v << 1, l, m, tl, tr) +
           get(v << 1 | 1, m + 1, r, tl, tr);
  }
};

void init() {
  // Khởi tạo các giá trị ngẫu nhiên
  for (int i = 0; i < maxn; i++) {
    rnd[i] = rng();
    H.insert(rnd[i]);
  }
}

void solve(int test) {
  int q, ans = 0;

  cout << "Case #" << test << ": ";
  cin >> n;
  SegmentTree segmentTree(n);

  for (int i = 1; i <= n; i++) {
    int p;
    cin >> p;
    segmentTree.update(1, 1, n, i, rnd[p]);
  }

  cin >> q;
  while (q--) {
    int type, l, r;
    cin >> type >> l >> r;

    if (type == 1) {
      segmentTree.update(1, 1, n, l, rnd[r]);
    } else {
      if (l == r) {
        ++ans;
        continue;
      }
      if ((r - l) % 2 == 1) continue;

      // Kiểm tra liệu f(l, m) - f(m + 1, r) có
      // thuộc tập H?
      int m = (l + r) / 2;
      ull L = segmentTree.get(1, 1, n, l, m),
          R = segmentTree.get(1, 1, n, m + 1, r);

      if (H.count(L - R)) {
```

[7] https://vnoi.info/wiki/algo/data-structures/segment-tree-basic.md

```
        ++ans;
        continue;
      }

      // Kiểm tra liệu f(m, r) - f(l, m - 1) có
      // thuộc tập H?
      L = segmentTree.get(1, 1, n, l, m - 1),
      R = segmentTree.get(1, 1, n, m, r);

      if (H.count(R - L)) {
        ++ans;
        continue;
      }
    }
  }

  cout << ans << '\n';
}

signed main() {
  ios_base::sync_with_stdio(false);
  cin.tie(0);
  cout.tie(0);

  init();

  int t;
  cin >> t;
  for (int i = 1; i <= t; i++) solve(i);
}
```

Bài toán 2 – HNOJ – trafficsystem [8]

Trước khi đến với phần lời giải, bạn đọc có thể thử sức với bài toán.

Tóm tắt đề bài

Một đất nước có $n \leq 10^5$ thành phố. Để di chuyển giữa 2 thành phố có 2 cách là sử dụng hệ thống đường bộ hoặc tàu điện ngầm (2 chiều). Ban đầu đất nước chưa có đường nối nào.

Cho $q \leq 10^5$ truy vấn thuộc hai loại sau:

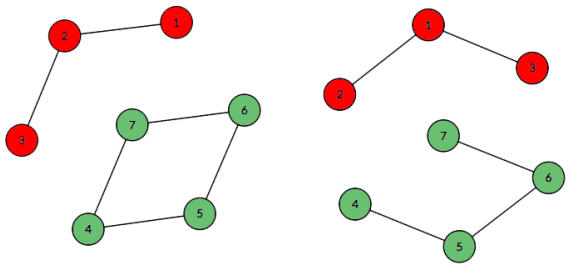
- $1\ u\ v$: Thiết lập đường bộ giữa 2 thành phố u và v.
- $2\ u\ v$: Thiết lập đường tàu điện ngầm giữa 2 thành phố u và v.

Sau mỗi truy vấn hãy cho biết hệ thống giao thông của đất nước có tốt không? In ra `YES` hoặc `NO` tương ứng với có hoặc không.

Hệ thống giao thông được gọi là tốt khi với mọi cặp thành phố $i < j$, nếu i đi được đến j qua đường bộ thì cũng phải đi được qua hệ thống tàu điện ngầm và ngược lại.

Nhận xét

Ta coi hệ thống đường bộ và tàu điện ngầm là 2 đồ thị riêng biệt. Coi mỗi thành phần liên thông là một tập hợp chứa các thành phố. Bài toán trở thành so sánh tập hợp A và B, trong đó A là tập hợp của các thành phần liên thông C_i đường bộ, B là tập hợp của các thành phần liên thông D_i tàu điện ngầm.



Trong hình ta có tập $A = \{\{1, 2, 3\}, \{5, 6, 7, 8\}\}$, $B = \{\{1, 2, 3\}, \{5, 6, 7, 8\}\}$. Như vậy có thể kết luận hệ thống này là tốt.

Lời giải

Với mỗi đỉnh ta sẽ gán một số nguyên khác không ngẫu nhiên. Với mỗi thành phần liên thông ta cần lưu giá trị Hash của các đỉnh thuộc tập đó, là XOR của tất cả giá trị được gán cho mỗi đỉnh. Việc này có thể dễ dàng xử lí được bằng CTDL Disjoint Set [9] .

Ta thu được 2 tập hợp $A = \{h(C_1), h(C_2), \ldots, (C_p)\}$ và $B = \{h(D_1), h(D_2), \ldots, h(D_q)\}$. Lực lượng của 2 tập vẫn có thể lên tới $O(n)$.

Một cách đơn giản là lấy tổng $S_A = h(C_1) + h(C_2) + \ldots + h(C_p)$ và $S_B = h(D_q) + h(D_q) + \ldots + h(D_q)$ rồi đem so sánh. Việc cập nhật S_A và S_B được thực hiện gộp 2 thành phần liên thông khi xử lí truy vấn.

```
#include <bits/stdc++.h>
using namespace std;

typedef unsigned long long ull const int maxn =
  100005

  ull r[maxn];
mt19937_64 rng(time(0));

struct DisjointSet {
  vector<int> parent;
  vector<ull> value;
  ull s;

  DisjointSet(int n, ull r[])
    : parent(n + 1), value(n + 1), s(0) {
    for (int i = 1; i <= n; i++) {
      parent[i] = i;
      value[i] = r[i];
      s += r[i];
    }
```

[8] https://hnoj.edu.vn/problem/trafficsystem
[9] https://vnoi.info/wiki/algo/data-structures/disjoint-set

```
  }

  int find(int u) {
    if (u != parent[u])
      return parent[u] = find(parent[u]);
    return u;
  }

  bool join(int u, int v) {
    u = find(u);
    v = find(v);
    if (u == v) return false;
    parent[u] = v;

    // Xóa 2 giá trị cũ
    s -= value[v] + value[u];

    // Gộp giá trị Hash của 2 thành phần liên
    // thông
    value[v] ^= value[u];

    // Thêm giá trị mới
    s += value[v];

    return true;
  }
};

signed main() {
  ios_base::sync_with_stdio(false);
  cin.tie(0);
  cout.tie(0);

  int n, q;
  cin >> n >> q;

  // Khởi tạo các giá trị ngẫu nhiên cho từng đỉnh
  for (int i = 1; i <= n; i++) { r[i] = rng(); }

  vector<DisjointSet> disjointSet(
    2, DisjointSet(n, r));

  while (q--) {
    int x, u, v;
    cin >> x >> u >> v;
    disjointSet[x - 1].join(u, v);

    if (disjointSet[0].s == disjointSet[1].s)
      cout << "YES\n";
    else
      cout << "NO\n";
  }
}
```

Lời bình

Trong cả hai bài toán trên, đều tồn tại lời giải bằng thuật toán tất định, việc này nhường lại cho bạn đọc.

Có thể thấy lời giải bằng Hash so với lời giải dùng thuật toán tất định rất dễ nghĩ ra chỉ là cải tiến từ hướng nghĩ của cách làm "trâu", cũng dễ cài, tốc độ thực thi nhanh.

Lưu trữ Hash

Trong rất nhiều trường hợp ta cần lưu trữ một lượng lớn các giá trị Hash để tiện cho việc kiểm tra, so sánh về sau. Việc lưu trữ một cách hiệu quả có thể giúp tránh việc mất điểm đáng tiếc hoặc cũng có thể tăng tốc các thuật toán chưa tối ưu (sub-optimal).

Thông thường, ta sẽ sử dụng các CTDL có sẵn để tiết kiệm thời gian. Để giúp bạn đọc lựa chọn CTDL phù hợp nhất cho lời giải của mình, dưới đây sẽ so sánh tốc độ của các CTDL sau trong các trường hợp khác nhau:

- `std::map`: Độ phức tạp các thao tác là $\mathcal{O}(\log n)$.
- `std::unordered_map`: Độ phức tạp trung bình $\mathcal{O}(\infty)$, trường hợp tệ nhất $\mathcal{O}(n)$.
- `__gnu_pbds::gp_hash_table`: Độ phức tạp trung bình $\mathcal{O}(1)$, trường hợp tệ nhất $\mathcal{O}(n)$.

`__gnu_pbds::gp_hash_table` là một CTDL tương đối "lạ", để sử dụng được CTDL này:

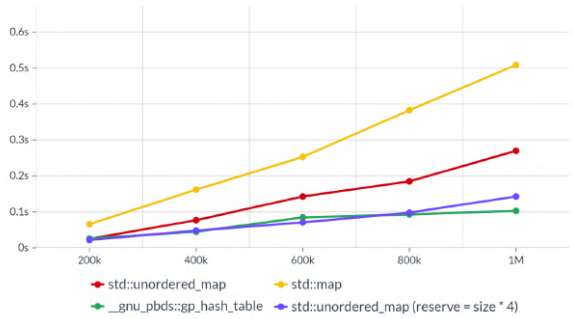
```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
using namespace __gnu_pbds;
using namespace std;

int main() {
  gp_hash_table<int, int> m;
  m[3] = 1;
  cout << m[3];
}
```

Tuy vậy cách dùng cũng không có nhiều khác biệt với 2 CTDL phổ biến là `std::map` và `std::unordered_map`.

Các bài kiểm tra dưới đây đều được dịch bằng C++14 trên hệ thống của Codeforces.

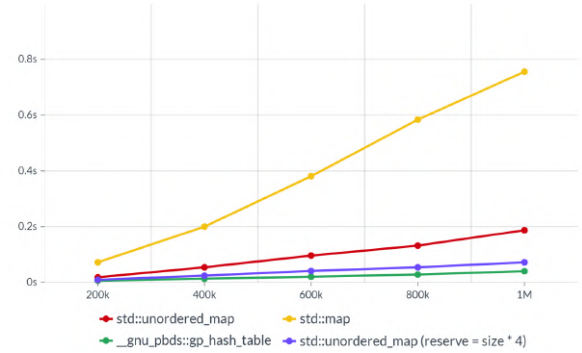
Tốc độ chèn

Chèn các phần tử là các số nguyên 64-bit ngẫu nhiên.



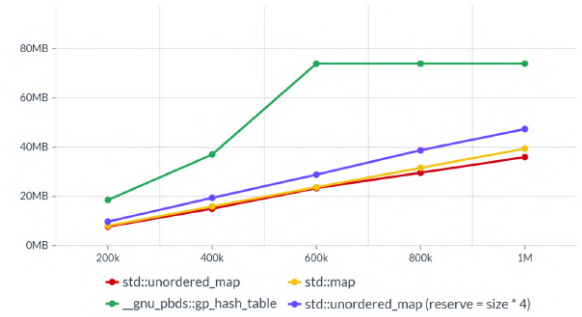
Tốc độ đọc

Đọc các phần tử là các số nguyên 64-bit ngẫu nhiên.



Bộ nhớ

Bộ nhớ (Megabyte) khi chèn các phần tử là các số nguyên 64-bit ngẫu nhiên.



Nhận xét

- `std::map`: Nhìn chung với tập dữ liệu ngẫu nhiên, `std::map` có thời gian thực thi lâu nhất. Ở đây ta lưu các giá trị Hash, tính thứ tự không quan trọng thì việc sử dụng `std::map` là lựa chọn không tối ưu.
- `__gnu_pbds::gp_hash_table`: Tốn tương đối nhiều bộ nhớ nhưng có tốc độ nhanh nhất. Một số trình biên dịch, ví dụ Clang, không sử dụng được CTDL này.
- `std::unordered_map`: Nếu ước lượng được số lượng key có thể thực hiện reserve bộ nhớ cho `std::unordered_map`, có thể giảm thời gian thực thi đáng kể, đổi lại là việc sử dụng nhiều bộ nhớ hơn một chút:

```
std::unordered_map<int, int> um;
um.reserve(SIZE * 4);
```

Nếu không thể ước lượng được số lượng key, có thể giảm load factor (tỉ lệ giữa số lượng phần tử có trong hash map và số lượng bucket) xuống khoảng 0.25, tùy trường hợp sẽ giúp giảm thời gian thực thi (tuy vậy sẽ sử dụng nhiều bộ nhớ hơn đáng kể):

```
um.max_load_factor(0.25);
```

Tốc độ của các CTDL này còn phụ thuộc vào hàm Hash, có thể tham khảo tại đây [10] . Tuy vậy trong hầu hết trường hợp không cần quan tâm vì hàm Hash mặc định vẫn chạy rất tốt trên tập dữ liệu ngẫu nhiên.

Bài kiểm tra trên mang tính chất tham khảo, do thời gian thực thi tùy thuộc vào bài toán và bộ test. Một cách để chọn CTDL phù hợp là sinh một số test lớn, chọn CTDL có thời gian chạy nhanh nhất.

Lời bạt

Có thể thấy Hash là kĩ thuật "nhỏ mà có võ" trong lập trình thi đấu.

Ngoài ra trong nhiều bài toán mà thuật toán chính để giải không phải là Hash, liên quan đến việc kiểm soát rất nhiều dữ liệu, Hash như một công cụ đắc lực giúp việc cài đặt nhanh và dễ hơn.

Hi vọng cả những nội dung về sinh số ngẫu nhiên và quản lí dữ liệu được trình bày ở trên cũng sẽ giúp ích cho hành trình lập trình thi đấu của bạn đọc!

Bài tập áp dụng

- Bedao Grand Contest 10 - PERFECT [11] (Nếu p là mảng các số nguyên không âm không quá n)
- Atcoder ABC250E - Prefix Equality [12]
- Hackerrank - Number Game on a Tree [13]
- Codeforces 1175F - The Number of Subpermutations [14]
- Codeforces 1418G - Three Occurrences [15]

[10] https://codeforces.com/blog/entry/62393
[11] https://oj.vnoi.info/problem/bedao_g10_perfect
[12] https://atcoder.jp/contests/abc250/tasks/abc250_e
[13] https://www.hackerrank.com/contests/hourrank-17/challenges/number-game-on-a-tree/editorial
[14] https://codeforces.com/problemset/problem/1175/F
[15] https://codeforces.com/contest/1418/problem/G
[16] https://en.cppreference.com/w/cpp/numeric/random/uniform_int_distribution

Đọc thêm

- `uniform_int_distribution`, một cách sinh số ngẫu nhiên khác từ C++11 trở đi [16]
- Sơ bộ về bảng băm [17]
- `__gnu_pbds::gp_hash_table` [18]

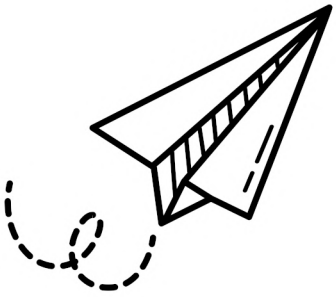
Tham khảo

- Codeforces [19]

[17] https://vnoi.info/wiki/algo/data-structures/hash-table.md
[18] https://usaco.guide/gold/faster-hashmap?lang=cpp
[19] https://codeforces.com/blog/entry/85900

Tối ưu quy hoạch động 1 chiều

Nguyễn Tuấn Tài
Sinh viên năm 1, Trường Đại học Khoa học Tự nhiên, Đại học Quốc gia TP.HCM



Giới thiệu: đây là kiến thức xuất hiện trong đề thi TST[20] 2022, và đã lấy đi rất nhiều nước mắt của thí sinh. Nếu bạn muốn thử học một thuật toán mới lạ mà nhiều người chưa biết (ngay cả Trần Xuân Bách!), thì đây chính là bài viết dành cho bạn!

Khi làm những bài toán quy hoạch động, đôi khi ta sẽ nghĩ ra những thuật toán có độ phức tạp rất lớn, ví dụ:

- $f[i][j] = \min_{0 \le k < j} f[i-1][k] + w(k,j)$ Công thức trên có độ phức tạp $O(n^3)$, có thể cải tiến xuống $O(n^2)$ hoặc $O(n^2 \log n)$ bằng quy hoạch động bao lồi/chia để trị (trong một số điều kiện nhất định).
- $f[i] = \min_{0 \le j < i} f[j] + w(j,i)$ Công thức trên có độ phức tạp $O(n^2)$, có thể cải tiến xuống $O(n \log n)$ hoặc $O(n)$ (trong một số điều kiện nhất định).

Ở bài viết này, chúng ta sẽ tìm hiểu về cách tối ưu công thức thứ 2, hay còn gọi là *phương pháp tối ưu quy hoạch động 1 chiều*.

Giới thiệu bài toán

Gọi $w(j,i)$ là một hàm tính cost thỏa mãn bất đẳng thức tứ giác (quadrangle inequality):

$$w(a,c) + w(b,d) \le w(a,d) + w(b,c)$$

với mọi $a < b \le c < d$. Ta sẽ tính toán công thức quy hoạch động sau với độ phức tạp nhanh hơn $O(n^2)$:

$$f[i] = \min_{0 \le j < i} f[j] + w(j,i)$$

Một số ví dụ về hàm w thỏa mãn bất đẳng thức tứ giác (bạn đọc có thể tự chứng minh):

- $w(j,i) = i - j$
- $w(j,i) = b[j] \cdot a[i]$ (với $a[j] \le a[i]$ và $b[j] \ge b[i]\ \forall\ j < i$)
- $w(j,i) = (a[i] - a[j])^2$ (với $a[j] \le a[i]\ \forall\ j < i$)

Thuật toán

Nhưng... làm sao để tối ưu công thức quy hoạch động trên? Có cách nào để nhanh chóng tìm được vị trí mà $f[j] + w(j,i)$ đạt giá trị nhỏ nhất không?

Ta định nghĩa mảng h như sau

$$h[i] = \arg\min_{0 \le j < i} f[j] + w(j,i)$$

Nói cách khác, $h[i]$ là vị trí j nhỏ nhất thỏa mãn $f[j] + w(j,i)$ đạt giá trị cực tiểu.

Để thuận tiện cho việc biểu diễn thuật toán, ta sẽ quy ước

$f[0] = 0,\ f[1] = f[2] = \ldots = f[n] = \infty.$

Ý tưởng "ngây thơ"

Trước khi đi vào thuật toán chính, chúng ta sẽ xem xét qua thuật toán "ngây thơ" sau:

- Ở thời điểm đầu tiên, $h[1] = h[2] = \ldots = h[n] = 0$.
- Ở thời điểm thứ i:
  - Vì $h[i]$ đã được cập nhật hoàn toàn, ta tính được $f[i] = f[h[i]] + w(h[i], i)$.
  - Sau khi tính được $f[i]$, ta sẽ cập nhật lại $h[i+1], h[i+2], \ldots, h[n]$.

Chúng ta có thể cài đặt thuật toán trên một cách đơn giản như sau:

```
const int N = 1e5 + 3;
int n, h[N];
long long f[N];

long long w(int j, int i) {
  // một hàm cost bất kì thỏa mãn
  // bất đẳng thức tứ giác
}

void solve() {
  for (int i = 1; i <= n; ++i) {
    // cập nhật f[i]
    f[i] = f[h[i]] + w(h[i], i);

    for (int j = i + 1; j <= n; ++j) {
      // cập nhật lại h[i + 1..n]
      if (f[i] + w(i, j) < f[h[j]] + w(h[j], j)) {
        h[j] = i;
      }
    }
  }
}
```

Ý tưởng chính

Dễ thấy thuật toán "ngây thơ" trên có độ phức tạp $O(n^2)$. Làm sao để cải tiến thuật toán? Liệu mảng h có một tính chất đặc biệt nào có thể giúp ta dễ dàng cập nhật được không?

[20] TST: Kỳ thi tuyển chọn đội tuyển dự thi Olympic Tin học Quốc tế, hay còn được biết đến là Vòng 2 thi chọn Học sinh giỏi Quốc gia.

Nhận xét †. *Ở mọi thời điểm, mảng h luôn là dãy đơn điệu tăng (tức $h[1] \le h[2] \le \ldots \le h[n]$).* (chứng minh ở phần Phụ lục)

Việc mảng h luôn là dãy đơn điệu tăng sẽ có ý nghĩa gì?

Khi cập nhật đến $f[i]$, nếu tồn tại một vị trí j thỏa mãn $f[h[j]] + w(h[j], j) \le f[i] + w(i, j)$, điều đó đồng nghĩa với việc $h[j]$ sẽ không thay đổi. Không chỉ thế, vì $h[i+1] \le h[i+2] \le \ldots \le h[j-1] \le h[j]$, nên cả đoạn $h[i+1 \ldots j]$ cũng sẽ không thay đổi.

Hệ quả. *Nếu tồn tại vị trí j thỏa mãn $f[h[j]] + w(h[j], j) \le f[i] + w(i, j)$, ta được $f[h[p]] + w(h[p], p) \le f[i] + w(i, p)$ với mọi $i < p \le j$.*

Chính vì thế, để cập nhật mảng h, ta sẽ tìm vị trí z nhỏ nhất thỏa mãn $f[h[z]] + w(h[z], z) > f[i] + w(i, z)$, và cập nhật $h[z] = h[z+1] = \ldots = h[n] = i$. Từ đây, ta có ý tưởng thuật toán sau:

Thuật toán.

Ta sẽ biểu diễn mảng h thành m đoạn $(l[i], r[i], p[i])$ thỏa mãn:

$$\begin{cases} l[1] = 1 \\ r[m] = n \\ p[i] = h[l[i]] = h[l[i]+1] = \ldots = h[r[i]] \\ l[i+1] = r[i] + 1, \ \forall\ 1 \le i < m \\ p[i] < p[i+1], \ \forall\ 1 \le i < m \end{cases}$$

- Ở thời điểm đầu tiên, mảng h chỉ chứa đoạn $(1, n, 0)$.
- Ở thời điểm thứ i:
  - Để tính được $f[i]$, ta sẽ tìm đoạn $(l[k], r[k], p[k])$ thỏa mãn $l[k] \le i \le r[k]$.
    - Vì mảng h đã cập nhật đến $f[i-1]$ nên $h[i]$ đã được cập nhật hoàn toàn (nhắc lại định nghĩa: $h[i] = \arg\min_{0 \le j < i} f[j] + w(j, i)$).
    - Lúc này, ta sẽ cập nhật $f[i] = f[p[k]] + w(p[k], i)$.
  - Sau khi tính được $f[i]$, ta sẽ cập nhật lại mảng h. Theo nhận xét phía trên, ta sẽ tìm vị trí z nhỏ nhất thỏa mãn $f[i] + w(i, z) < f[h[z]] + w(h[z], z)$. Xét đoạn cuối cùng trong mảng h, giả sử đoạn đó là (l, r, p).
    - Nếu $f[i] + w(i, l) < f[p] + w(p, l)$, ta sẽ xóa đoạn đó khỏi mảng h và tiếp tục xét đoạn cuối cùng tiếp theo.
    - Ngược lại, ta sẽ tìm vị trí z nhỏ nhất thỏa mãn $f[i] + w(i, z) < f[p] + w(p, z)$ bằng tìm kiếm nhị phân, sau đó cập nhật lại (l, r, p) thành $(l, z-1, p)$, và thêm đoạn (z, n, i) vào cuối mảng h.

Cài đặt mẫu

Ta có thể cài đặt thuật toán trên bằng cách sử dụng deque. Để thuận tiện cho việc cài đặt, ta sẽ không lưu lại các giá trị $h[i]$ đã qua sử dụng.

```
struct item {
  int l, r, p;
};

const int N = 1e5 + 3;
int n;
long long f[N];

long long w(int j, int i) {
  // một hàm cost bất kì thỏa mãn
  // bất đẳng thức tứ giác
}

void solve() {
  deque<item> dq;
  dq.push_back({1, n, 0});
  for (int i = 1; i <= n; ++i) {
    f[i] = f[dq.front().p] + w(dq.front().p, i);
    // deque chỉ lưu giá trị từ h[i + 1]
    // tới h[n]
    ++dq.front().l;

    // nếu l > r, ta loại đoạn này khỏi deque
    if (dq.front().l > dq.front().r) {
      dq.pop_front();
    }

    while (!dq.empty()) {
      auto [l, r, p] = dq.back();
      if (f[i] + w(i, l) < f[p] + w(p, l)) {
        dq.pop_back();
        // p không còn là giá trị của
        // h[l], h[l + 1], ..., h[r]
        // lúc này, h[l]=h[l+1]=...=h[r]=i.
      } else
        break;
    }

    if (dq.empty()) {
      dq.push_back({i + 1, n, i});
      // h[i+1]=h[i+2]=...=h[n]=i
    } else {
      // tìm nhị phân vị trí pos nhỏ nhất
      // thỏa mãn h[pos] = i
      auto &[l, r, p] = dq.back();
      int low = l, high = r;
      int pos = r + 1, mid;
      while (low <= high) {
        mid = (low + high) / 2;
        if (f[i] + w(i, mid) < f[p] + w(p, mid)) {
          pos = mid, high = mid - 1;
        } else {
          low = mid + 1;
        }
      }

      // cập nhật đoạn (l,r,p) thành (l,pos-1,p)
      r = pos - 1;
      if (pos <= n) {
        dq.push_back({pos, n, i});
        // h[pos]=h[pos+1]=...=h[n]=i
      }
    }
  }
}
```

Vì số đoạn tối đa được thêm vào deque trong cả quá trình là n, kết hợp với tìm kiếm nhị phân ở cuối mỗi thời điểm, ta được độ phức tạp cuối cùng là $O(n \log n)$.

Bài toán 1

Codeforces[21] - 319C [22]

Tóm tắt

Cho n cây được đánh số hiệu từ 1 tới n, mỗi cây có độ cao lần lượt là $a_1, a_2, \ldots, a_n$. Alob và Bice muốn cưa đổ tất cả n cây sao cho tốn ít chi phí nhất.

Alob và Bice có một cái cưa máy, mỗi lần sử dụng cưa có thể giảm độ cao của một cây bất kì xuống 1. Tuy nhiên, sau mỗi lần sử dụng, cưa máy cần được sạc lại. Chi phí để sạc phụ thuộc vào những cây đã được chặt hoàn toàn (những cây đã được giảm độ cao về 0): trong những cây đã được chặt hoàn toàn, giả sử cây có số hiệu lớn nhất là i, chi phí để sạc cưa máy là b_i. Nếu không có cây nào đã được chặt hoàn toàn, ta không thể sạc lại cưa máy.

Điều kiện bài toán:

$$\begin{cases} 1 \le n \le 10^5 \\ 1 = a_1 < a_2 < \ldots < a_n \le 10^9 \\ 10^9 \ge b_1 > b_2 > \ldots > b_n = 0 \end{cases}$$

Ý tưởng

Vì $b_n = 0$, ta sẽ tìm chi phí nhỏ nhất để chặt hoàn toàn cây n (sau đó, ta có thể chặt bất kì cây nào mà không tốn chi phí).

Gọi $f[i]$ là chi phí nhỏ nhất để chặt hoàn toàn cây thứ i. Nếu cây gần nhất được chặt hoàn toàn trước đó là j, chi phí nhỏ nhất để chặt hoàn toàn cây thứ i sẽ là $f[j] + b_j \cdot a_i$. Vì vậy, ta có được công thức quy hoạch động sau:

$$f[i] = \min_{1 \le j < i} f[j] + b_j \cdot a_i$$

Nếu đặt $w(j, i) = b_j \cdot a_i$, hàm w là một hàm thỏa mãn bất đẳng thức tứ giác.

Chứng minh.

Xét 4 điểm $x < y \le z < t$, ta có:

$$\begin{aligned} & w(x,z) + w(y,t) - w(x,t) - w(y,z) \\ = \; & b_x \cdot a_z + b_y \cdot a_t - b_x \cdot a_t - b_y \cdot a_z \\ = \; & (b_x - b_y)(a_z - a_t) \le 0 \end{aligned}$$

Vì vậy,

$$w(x,z) + w(y,t) \le w(x,t) + w(y,z)$$

Từ đây, ta có thể áp dụng thuật toán đã nêu trong bài.

Cài đặt mẫu

```cpp
#include <bits/stdc++.h>
using namespace std;

struct item {
  int l, r, p;
};

const int N = 1e5 + 3;
int n, a[N], b[N];
long long f[N];

long long w(int j, int i) {
  return 1LL * b[j] * a[i];
}

void solve() {
  deque<item> dq;
  dq.push_back({2, n, 1});
  for (int i = 2; i <= n; ++i) {
    f[i] = f[dq.front().p] + w(dq.front().p, i);
    ++dq.front().l;
    if (dq.front().l > dq.front().r) {
      dq.pop_front();
    }

    while (!dq.empty()) {
      auto [l, r, p] = dq.back();
      if (f[i] + w(i, l) < f[p] + w(p, l)) {
        dq.pop_back();
      } else
        break;
    }

    if (dq.empty()) {
      dq.push_back({i + 1, n, i});
    } else {
      auto &[l, r, p] = dq.back();
      int low = l, high = r, pos = r + 1, mid;
      while (low <= high) {
        mid = (low + high) / 2;
        if (f[i] + w(i, mid) < f[p] + w(p, mid)) {
          pos = mid, high = mid - 1;
        } else {
          low = mid + 1;
        }
      }

      r = pos - 1;
      if (pos <= n) { dq.push_back({pos, n, i}); }
    }
  }
}

int main() {
  cin >> n;
  for (int i = 1; i <= n; ++i) { cin >> a[i]; }
  for (int i = 1; i <= n; ++i) { cin >> b[i]; }
  solve();
  cout << f[n];
}
```

Bài toán 2

VNOJ - Lễ hội [23]

[21] Codeforces: một trang web tổ chức các cuộc thi lập trình thi đấu với các dạng đề, bài tập đa dạng.

[22] https://codeforces.com/contest/319/problem/C

[23] https://oj.vnoi.info/problem/lehoi

Tóm tắt

Ở ngôi làng nọ, có n ngôi nhà nằm trên một đường thẳng. Biết trưởng làng sống ở nhà thứ 1, nhà thứ i nằm cách nhà trưởng làng đúng a_i km về phía đông. Trưởng làng muốn chọn ra một số địa điểm trên đường thẳng để chuẩn bị tổ chức lễ hội, biết chi phí tổ chức lễ hội cho một địa điểm là k.

Sau khi chuẩn bị xong, tất cả người dân sẽ di chuyển đến địa điểm gần nhà mình nhất để tham gia lễ hội. Nếu một người dân ở cách địa điểm tổ chức x km, chi phí để di chuyển sẽ là x.

Hãy tìm cách chọn một số địa điểm sao cho tổng chi phí tổ chức lễ hội và di chuyển là nhỏ nhất.

Nói cách khác, nếu như ta chọn m địa điểm, địa điểm thứ i nằm cách nhà trưởng làng đúng s_i km về phía đông, tổng chi phí tổ chức lễ hội và di chuyển sẽ là $k \cdot m + \sum_{i=1}^{n} \min_{j=1}^{m} |a_i - s_j|$.

Điều kiện bài toán:

$$\begin{cases} 1 \le n \le 2 \cdot 10^5, 1 \le k \le 10^9 \\ 0 = a_1 < a_2 < \ldots < a_n \le 10^9 \end{cases}$$

Ý tưởng

Ta có nhận xét sau tất cả người dân nằm trên một đoạn liên tiếp sẽ đến cùng một địa điểm, vì thế bài toán có thể viết lại thành chia n người dân thành các đoạn liên tiếp sao cho tổng chi phí là nhỏ nhất, biết chi phí mỗi đoạn gồm chi phí tổ chức k và chi phí di chuyển của người dân trong đoạn.

Gọi $f[i]$ là chi phí nhỏ nhất để chia i người dân thành các đoạn sao cho tổng chi phí là nhỏ nhất. Ta có công thức quy hoạch động sau:

$$f[i] = k + \min_{0 \le j < i} f[j] + w(j, i)$$

với $w(j, i)$ là chi phí di chuyển của người dân nằm trong đoạn $j + 1$ tới i.

Ta sẽ chứng minh hàm w thỏa mãn bất đẳng thức tứ giác.

Chứng minh.

Đặt số người dân trong đoạn $[l + 1, r]$ là $t = r - l$.

Đặt $p[i] = a_1 + a_2 + \ldots + a_i$. Ta có 2 trường hợp sau:

- Nếu t chẵn, phương án đặt địa điểm tập trung tối ưu nhất là ở giữa người thứ $\frac{t}{2}$ và người thứ $\frac{t}{2} + 1$. Chi phí di chuyển trong trường hợp này là $\sum_{i=r-\frac{t}{2}+1}^{r} a_i - \sum_{i=l+1}^{l+\frac{t}{2}} a_i$, hay $(p[r] - p[r - \frac{t}{2}]) - (p[l + \frac{t}{2}] - p[l])$.
- Ngược lại, phương án đặt địa điểm tập trung tối ưu nhất là ở nhà của người thứ $\frac{t+1}{2}$. Chi phí di chuyển trong trường hợp này là $\sum_{i=r-\frac{t-1}{2}+1}^{r} a_i - \sum_{i=l+1}^{l+\frac{t-1}{2}} a_i$, hay $(p[r] - p[r - \frac{t-1}{2}]) - (p[l + \frac{t-1}{2}] - p[l])$.

Xét 4 điểm $x < y \le z < t$. Để thuận tiện cho việc chứng minh, ta sẽ giả sử độ dài các đoạn đều là chẵn. Các trường hợp còn lại cũng có thể chứng minh tương tự.

Với $r - l$ chẵn: $w(l, r) = (p[r] - p[r - \frac{k}{2}]) - (p[l + \frac{k}{2}] - p[l]) = p[l] + p[r] - 2 \cdot p[\frac{l+r}{2}]$.

Đặt $b = \frac{x+z}{2}$, $c = \frac{y+t}{2}$, $d = \frac{x+t}{2}$, $e = \frac{y+z}{2}$. Ta có:

$$\begin{aligned} & w(x,z) + w(y,t) - w(x,t) - w(y,z) \\ = \; & 2 \cdot (-p[\frac{x+z}{2}] - p[\frac{y+t}{2}] + p[\frac{x+t}{2}] + p[\frac{y+z}{2}]) \\ = \; & 2 \cdot (-p[b] - p[c] + p[d] + p[e]) \\ = \; & 2 \cdot (p[d] - p[b]) - 2 \cdot (p[c] - p[e]) \\ = \; & 2 \cdot (a_{b+1} + a_{b+2} + \ldots + a_d) \\ & -2 \cdot (a_{e+1} + a_{e+2} + \ldots + a_c) \\ = \; & 2 \cdot (a_{b+1} - a_{e+1}) + 2 \cdot (a_{b+2} - a_{e+2}) + \ldots \\ & +2 \cdot (a_d - a_c) \le 0 \end{aligned}$$

Vì vậy,

$$w(x, z) + w(y, t) \le w(x, t) + w(y, z)$$

Cài đặt mẫu

```
#include <bits/stdc++.h>
using namespace std;

struct item {
  int l, r, p;
};

const int N = 2e5 + 3;
int n, k, a[N];
long long p[N], f[N];

long long w(int l, int r) {
  int t = (r - l) / 2;
  return (p[r] - p[r - t]) - (p[l + t] - p[l]);
}
```

```
void solve() {
  deque<item> dq;
  dq.push_back({1, n, 0});
  for (int i = 1; i <= n; ++i) {
    f[i] =
      k + f[dq.front().p] + w(dq.front().p, i);
    ++dq.front().l;
    if (dq.front().l > dq.front().r) {
      dq.pop_front();
    }

    while (!dq.empty()) {
      auto [l, r, p] = dq.back();
      if (f[i] + w(i, l) < f[p] + w(p, l)) {
        dq.pop_back();
      } else
        break;
    }

    if (dq.empty()) {
      dq.push_back({i + 1, n, i});
    } else {
      auto &[l, r, p] = dq.back();
      int low = l, high = r, pos = r + 1, mid;
      while (low <= high) {
        mid = (low + high) / 2;
        if (f[i] + w(i, mid) < f[p] + w(p, mid)) {
          pos = mid, high = mid - 1;
        } else {
          low = mid + 1;
        }
      }

      r = pos - 1;
      if (pos <= n) { dq.push_back({pos, n, i}); }
    }
  }
}

int main() {
  cin >> n >> k;
  for (int i = 1; i <= n; ++i) {
    cin >> a[i];
    p[i] = p[i - 1] + a[i];
  }
  solve();
  cout << f[n];
}
```

Phụ lục

Chứng minh nhận xét †.

Ta sẽ chứng minh bằng cách phản chứng: giả sử tồn tại vị trí i thỏa mãn $h[i] > h[i+1]$. Để thuận tiện cho việc chứng minh, ta sẽ đặt $a = h[i],\ b = h[i+1]$ $(a > b)$. Điều này tương đương với:

$$\begin{cases} f[a] + w(a,i) < f[b] + w(b,i) \\ f[a] + w(a,i+1) > f[b] + w(b,i+1) \end{cases}$$

Trừ hai bất đẳng thức theo vế, ta được:

$$w(a,i) - w(a,i+1) < w(b,i) - w(b,i+1)$$

$$\Leftrightarrow w(a,i) + w(b,i+1) < w(a,i+1) + w(b,i)$$

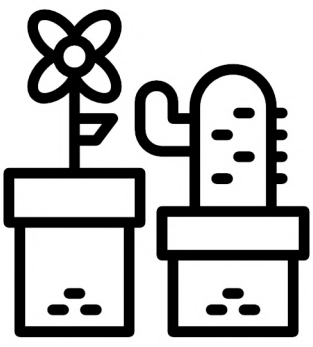
Tuy nhiên, theo tính chất của hàm w, xét bộ số $b < a < i < i+1$, ta có:

$$w(b,i) + w(a,i+1) \leq w(b,i+1) + w(a,i)$$

Điều này là vô lý. Vì vậy, ta có điều phải chứng minh.

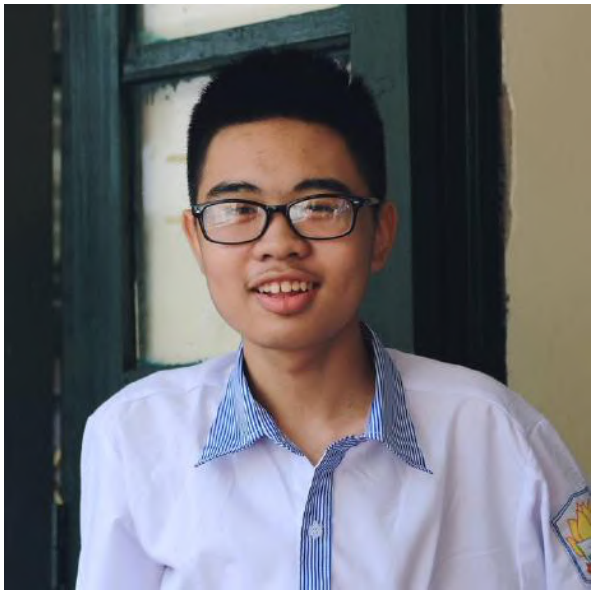
Phỏng vấn Bùi Việt Dũng

Interviewer: Vương Hoàng Long – ICPC World Finalist 2021



Long: Chào Dũng, rất cảm ơn bạn đã nhận lời phỏng vấn cho tạp chí VNOI năm đầu tiên này. Bạn cũng là người đầu tiên bọn mình phỏng vấn . Chúng ta có thể bắt đầu nhé, bạn có thể giới thiệu qua một số thông tin cá nhân được không?

Dũng: Mình tên là Bùi Việt Dũng, năm nay đang 22 tuổi, là cựu học sinh trường THPT Kim Liên, Hà Nội. Mình tham gia ICPC[24] 4 năm với tư cách là sinh viên của trường Đại học Wisconsin-Madison... tạm thời là thế nhé! .



Bùi Việt Dũng tại trường THPT Kim Liên

Long: Sở thích của bạn là gì nhỉ? Bạn thường làm gì lúc rảnh?

Dũng: Sở thích thì mình cũng có nhiều. Mình thích xem nhiều chương trình truyền hình, một phần là để luyện tiếng Anh, phần chính là để giải trí.

Long: Show yêu thích của bạn là show nào?

Dũng: Arthur, một show của thiếu nhi. Đặc biệt cái show này có hai lớp nghĩa, một lớp là để trẻ con xem thì hiểu, một lớp khác là để người lớn xem lại hiểu theo một nghĩa khác. Mình xem show này từ lớp 4, sau này mình xem lại thì lại hiểu khác, cho nên mình thấy khá là thú vị ;)

Long: Thú vị, thú vị đấy! Bạn đang ở Chicago đúng không nhỉ? Bạn thích điều gì ở Chicago nhất?

Dũng: Một điều mà mình yêu thích ở Chicago là mọi người đều đặt hàng online, và hàng cũng đến rất nhanh. Ở Chicago thì có ngay một kho hàng ở đây, Chicago cũng là một trong những trung tâm thương mại lớn của thế giới. Nếu các bạn biết về thị trường, về quyền mua, quyền bán ấy, như CMI các thứ thì Chicago có đủ các kho hàng trên đời. Nói chung là mua gì cũng đến rất nhanh, nhiều khi sáng mình đặt là chiều đã có rồi!

Long: Hỏi ngoài lề tí, bạn thích mua hàng online hay bạn thích đến tận nơi để mua? Vừa rồi Black Friday bạn có mua nhiều hàng không?

Dũng: Black Friday thì mình toàn mua hàng online vì mình không thích chen lấn xô đẩy lắm. Một trong số thứ mình mua là chiếc laptop mình đang xài đây, chất lượng hình ảnh nó cũng khá cao, ít nhất là cao hơn hồi mình làm bvd live trên VNOI.

Long: Okay, thế bạn mua máy gì nhỉ? Thinkpad? Macbook?

Dũng: Mình chơi hẳn một cái Alienware luôn, gaming PC nhé.

Long: Uii. Alienware là đỉnh của việc chơi game rồi!

Dũng: Đã thế máy này là mình còn được giảm tận 60%.

Long: Thế bạn có chơi game không?

Dũng: Mình không chơi game nhưng mình chỉnh sửa video khá nhiều. Ngoài tham gia cộng đồng Tin học mình còn tham gia một cộng đồng về tiếng Anh, ở đấy mình cũng làm bvd live but in English .

Long: Okay thế mình sẽ quay lại những câu hỏi chính. Mình sẽ muốn bạn nói theo trình tự thời gian như từ cấp 2 lên cấp 3 rồi lên đại học chẳng hạn, để độc giả dễ theo dõi. Ở cấp 2 thì bạn có học môn chuyên nào không? Hay là chỉ học đều các môn?

Dũng: Cấp 2 thì mình học ở trường THCS Trưng Vương, Hà Nội và mình học lớp chuyên Toán 2 của trường. Nói thế chứ mình duy trì cái chuyên Toán đến giữa lớp 7 thôi, chỉ học đến cách chia đa thức các thứ... Rồi sau đấy mình chuyển sang học Tin. Lẽ ra mình nên học chuyên Toán lâu hơn một chút, nó có thể sẽ giúp ích hơn. Hồi đấy mình tình cờ đọc được quyển sách giáo khoa Tin học lớp 11, có hướng dẫn lập trình trong Pascal khá đầy đủ, thế là mình chuyển sang tìm hiểu về Tin luôn .

Long: Sau đó thì bạn vào trường THPT Kim Liên, theo mình biết thì trường đấy không phải là trường chuyên ở Hà Nội nhưng bạn vẫn vào được đội tuyển và thi học sinh giỏi. Bạn có thể kể về quá trình bạn học ở cấp 3 được không? Làm sao mà bạn duy trì được việc học Tin trong một môi trường không chuyên như vậy?

Dũng: Mình bắt đầu thi Tin và có nền tảng từ cấp 2 rồi. Đến năm lớp 10 thì vì mình đã có giải

[24] ICPC: International Collegiate Programming Contest - Cuộc thi Lập trình Quốc tế lâu đời và danh giá nhất dành cho sinh viên các trường đại học và cao đẳng trên toàn cầu.

Nhì thành phố ở lớp 9 nên được chọn luôn vào đội tuyển của trường Kim Liên. Ở Kim Liên thì ít học sinh trong đội tuyển Tin nên không có thi chọn. Năm đấy thì cũng có các anh chị 11, 12 cũng có kinh nghiệm thi Tin từ trước và trường cũng muốn đầu tư nên đã mời cô Chinh ở Chuyên Sư Phạm về dạy, nhưng khá buồn là cô chỉ dạy được năm lớp 10 đấy thôi.

Dũng: Tuy cô Chinh chỉ dạy được một số buổi năm lớp 10 nhưng việc đó cũng tạo cho mình khá nhiều động lực học và có một số hướng đi nhất định. Mình sau đó cũng có động lực để giải bài và chỉnh lại hướng học của mình, ví dụ như hồi lớp 10 mình có lên group VNOI trên Facebook hỏi về những vấn đề như Treap, Luồng cực đại, Cây khung nhỏ nhất,... những cái mà mình tưởng mình hiểu rồi vì thuật toán đơn giản nhưng thực ra nếu yêu cầu mình chứng minh vì sao bước này trong thuật lại đúng, tại sao mình làm bước này bước kia thì mình không làm được. Mình nghĩ chính cái việc chứng minh được các thuật toán mới giúp các bạn thực sự hiểu về thuật toán và biết cách áp dụng nó vào giải các bài toán khó. Đấy là điều mình thấy bổ ích nhất khi mình được học với cô Chinh.

Long: Thế khi mà không học với cô Chinh nữa thì bạn tiếp tục học như thế nào?

Dũng: Hồi đấy thì hình như VNOI Wiki chưa có nhiều thứ như bây giờ. Mình thì bắt đầu học lý thuyết ở bộ 3 quyển tài liệu giáo khoa chuyên Tin. Ngoài ra mình còn đọc một số quyển khác như Một số vấn đề trong Tin học, KCBOOK,... tài liệu thì mình cũng chỉ có thế thôi.

Long: KCBOOK là một quyển sách rất là hay, lâu lắm rồi mình mới nghe thấy nó được nhắc đến đấy!

Dũng: Còn về việc giải bài thì hồi đầu lớp 10 mình chưa tìm được bản pdf của quyển Competitive Programming 3 nhưng đã có danh sách bài tập của quyển đấy trên UVa rồi. Thế nên mình cứ làm theo cái danh sách đấy, kiểu mình thấy chủ đề nào mình mở một vài bài ra mình không làm được thì mình sẽ tập trung vào chủ đề đấy, hoặc chủ đề nào mình thấy lạ lạ thì làm.

Dũng: Đến hồi hè lớp 10 thì mình biết đến Kattis, ở đấy có nhiều bài dạng ICPC vì Kattis cũng có nhiều đề ICPC các nơi trên thế giới. Mình thấy giải bài trên đấy rất hữu ích vì các đề ICPC hầu như đều có lời giải, và mình có thể đọc lời giải khi bí. Mình thấy giai đoạn này trình mình lên nhanh nhất. Sau này thì mình cũng không nhớ là mình có tập trung giải bài trên site nào khác nữa .

Long: Rất thú vị, mình để ý là bạn không hề nhắc đến Codeforces[25] trong khi bên trường Chuyên KHTN thì lại giải Codeforces rất nhiều.

Dũng: À, mình cũng có đụng đến Codeforces và cả Codechef nữa nhưng chỉ dùng để thi chứ không luyện bài trên đấy. Thật ra mình nghĩ giải bài ở đâu thì cũng tốt, nhưng cá nhân mình thấy hồi lớp 10 mình giải bài thì những bài trên Kattis chất lượng khá cao, mình làm thì thấy khá là trí tuệ . Thuật toán thì đơn giản nhưng nghĩ ra được thì khó.

Long: Thế thì sau khi bạn tự học như thế thì quá trình bạn thi thế nào nhỉ?

Dũng: Hồi mình lớp 10 thì may mắn là cả đội tuyển của trường mình đều vào được vòng 2 thành phố. Vòng 1 của lớp 10 thì mình thấy độ khó tương đương với thi thành phố ở lớp 9 hiện nay, nhưng khi lên lớp 11 thì vòng 1 có vẻ độ khó tăng lên rất nhiều. Hình như là thi thành phố ở Hà Nội vẫn duy trì cái độ khó đấy. Lúc mình lớp 11 thì mình khá phát triển rồi, cũng có nick xanh nên việc tăng độ khó như thế cũng không ảnh hưởng gì đến kết quả của mình. Thậm chí là mình thấy điểm của mình ở hai năm thì khá gần nhau, lớp 10 thì 16đ, còn lớp 11 thì 16.5đ, nhưng rank lớp 11 của mình cao hơn rất nhiều so với lớp 10.

Long: Đề Hà Nội mấy năm nay thì mình cũng không rõ, đội Kim Liên năm ngoái có một bạn nick xanh mà cũng không qua được vòng 1, nên chắc là hồi của mình thì đề nó không kinh khủng như bây giờ. Mình chỉ tiếc là mình chưa bao giờ qua được vòng 2 để vào được đội tuyển thi VOI, có lẽ đó là điều đáng tiếc nhất trong hành trình học Tin cấp 3 của mình.

Dũng: Thế thì bạn có tham gia thi vòng 2 ở Hà Nội 2 năm đúng không nhỉ? Các năm đấy bạn không vượt qua là vì đề khó hay có sự cố gì xảy ra không?

Long: Năm lớp 10 thì mình đọc đề rồi nghĩ ra được thuật giải của 2 bài, mà bài thứ 2 lại là bài hình học, mình thì lại chưa biết phương trình đường thẳng nên mình không code được. Thế là mình chỉ làm được 1 bài và kết quả mình thì không đủ để đi tiếp. Năm lớp 11 thì mình làm được 2 bài, bài còn lại thì impossible. Mình nghĩ bài 2 là một bài cặp ghép vì lúc học về cặp ghép mình đã gặp bài tương tự rồi. Chiến thuật lúc đó là mình bỏ bài 2, mình dành 2 tiếng đầu làm bài 1 và bài 3 rồi dành tiếng cuối để sinh test rồi stress test, mình cũng dựa trên post tự viết trình chấm của VNOI Wiki mà làm. Mình đã chắc chắn 100% là mình

[25] Codeforces: một trang web tổ chức các cuộc thi lập trình thi đấu với các dạng đề, bài tập đa dạng.

AC cả 2 bài đấy, nhưng kết quả thực tế thì mình chỉ được 1 nửa như thế, mà vòng 2 thì không có phúc khảo nên mình cũng không làm được gì hơn nên mình cũng đành chấp nhận kết quả như thế.

Dũng: Dù sao thì trong Tin học nhiều khi bạn nghĩ ra lời giải nhưng khi mình code ra thì chưa chắc nó đã ăn điểm giống như mình muốn. Ví dụ như World Finals[26] vừa rồi cũng có một bài hình, mình nghĩ ra được lời giải đúng y hệt trong video của giám khảo, cũng code ra y hệt vậy nhưng nộp lên chỉ được Wrong Answer hoặc code chạy quá thời gian thôi. Nói chung việc nghĩ ra được thuật giải là một chuyện, và chuyển hóa nó thành code và ăn được điểm lại là câu chuyện khác.

Bùi Việt Dũng tại ICPC World Finals 2021, Dhaka

Long: Mình đồng ý điều này . Okay thì đó là cấp 3 của bạn, sau đó thì hình như bạn không học đại học ở Việt Nam năm nào đâu nhỉ?

Dũng: Đúng rồi, mình du học Mỹ từ lớp 12. Đó là lý do vì sao mình chỉ thi vòng 2 lớp 10 và 11 còn lớp 12 tự dưng mình biến mất. Cho nên là năm lớp 11 mình rất quyết tâm là phải vào được đội tuyển thi VOI, còn không thì không bao giờ có cơ hội nữa luôn .

Long: Interesting, thế thì sang Mỹ bạn có làm USACO các thứ không?

Dũng: Mình có thử USACO nhưng vấn đề là bên Mỹ là số lượng học sinh làm CP[27] nhiều hơn Việt Nam rất nhiều, cho nên có nhiều bạn nick đỏ cạnh tranh để vào được vòng TST[28] của Mỹ, họ cũng chỉ lấy 16 người thôi. Cuối cùng là mình cũng không có cửa luôn rồi .

Long: Thế thì lên đại học bạn lại quyết định chọn UW-Madison?

Dũng: Vì đấy là trường rank cao nhất trong danh sách mà mình đỗ được, với lại thêm lý do nữa là trước khi apply thì mình cũng tạo một cái script quét qua lịch sử thi ICPC của các trường trong khu vực rồi sau đó có bảng rank các trường về thành tích ICPC nữa. UW-Madison thì chỉ lỡ ICPC World Finals có 3 lần và chưa lỡ lần nào trong thế kỷ 21 nên trường đó cũng có rank khá cao về phía ICPC, ngang ngửa MIT và Harvard luôn.

Long: Interesting, sau khi vào UW-Madison thì bạn thi ICPC thế nào nhỉ?

Dũng: Mình thi ICPC từ năm nhất luôn, năm đấy thì mình cũng may mắn nằm trong đội 1 của trường nhưng đến lúc thi Regional thì lại bị thua penalty team khác. Thật ra cũng có một sự cố khác xảy ra, đó là trình dịch C++ bị lỗi, các bạn thử tưởng tượng thi ICPC mà không có C++ xem .

Dũng: Nhưng rất may vì có sự cố đó mà 3 năm sau thi Regional thì kỳ thi được tổ chức tốt hơn rất nhiều. Ở máy có đủ mọi loại editor và compiler, chỉ tiếc mỗi năm đầu lại không được như thế, nhưng mà giả sử mình được đi WF vào năm nhất với năm hai thì mình lại không có cơ hội đi năm ba, năm tư nữa. Chưa kể là năm hai của mình (2019) trở về sau là có thêm North America Championship (NAC) nữa, nên là có lẽ trượt ở năm nhất đã tạo cơ hội cho mình thi ICPC nhiều hơn và trình độ của mình thì càng được nâng cao hơn. Thực ra bây giờ thì mình không thấy tiếc nữa, chỉ tiếc là năm nhất thì 2 teammate của mình lại là năm cuối nên họ không có cơ hội thi tiếp nữa, nhưng thôi chuyện đã qua rồi thì cũng không thể làm lại được .

Long: Okay, thế năm hai là bạn đã vào được WF chưa?

Dũng: Đúng rồi, năm đó thi Regional team

[26]World Finals: Cuộc thi ICPC cuối cùng trong một mùa giải bao gồm những đội xuất sắc nhất của các trường đại học trên thế giới vượt qua vòng loại vùng (Regional contest)

[27]CP: Competitive Programming - Lập trình thi đấu

[28]TST: Kỳ thi tuyển chọn đội tuyển dự thi Olympic Tin học Quốc tế, hay còn được biết đến là Vòng 2 thi chọn Học sinh giỏi Quốc gia.

[29]IOI: Kỳ thi Olympic Tin học Quốc tế.

mình thắng team thứ hai đến 3 bài. Team của mình có một bạn Trung Quốc nick đỏ và một bạn Thái Lan có huy chương đồng IOI[29] .

Q Năm đó bạn thi Regional ở site nào nhỉ?

Dũng: North Central North America (NCNA).

Long: Năm đó bạn vô địch Regional rồi thì vô thẳng WF nhỉ?

Dũng: Đúng rồi, nhưng team mình vẫn được mời tham gia thi unofficial ở NAC. Cũng hơi tiếc vì đó cũng là kỳ thi onsite đầu tiên của mình, mình cũng khá choáng ngợp. Lúc thi thì team mình performance cũng không tốt lắm, mình nhớ là top 17 thì sẽ được 1500$ và team mình thì lại xếp thứ 18 .

Dũng: Mình tiếc là hôm đấy có bài C là bài hình học, mà mình lại phụ trách phần hình học trong team nhưng mình lại chỉ nhớ mang máng cách làm mà lại không làm được hẳn. Ngoài ra, có một bài nữa mà teammate mình từng gặp ở APIO[30] rồi, chỉ là giới hạn khác một chút và thuật của teammate mình thì lại bị TLE. Có thêm bài A là một bài 2-SAT, thế mà teammate mạnh nhất của mình lại đọc sai đề nên thọt luôn bài đấy.

Long: Sau đó là WF năm đó diễn ra ở đâu nhỉ?

Dũng: Ở Nga, lúc đó mình cũng trên tinh thần làm lại bằng WF rồi nhưng lại dính COVID mới bùng phát nên là WF (bảng Invitational) lại bị delay đến năm 2021. Mình nhớ là mình qualify cho kỳ WF tiếp theo thì kỳ WF trước mới diễn ra

Dũng: COVID diễn ra khiến timeline của ICPC nó đảo lộn hết tất cả. Đã có những lúc mà mình phải train ở 2 team khác nhau cùng lúc, một đội để qualify năm sau và một đội cho năm ngoái.

Long: Nghe rất là hardcore. Năm sau (năm 2023) thì WF lại tiếp tục diễn ra vào tháng 11 và nó sẽ là 2 kỳ thi gộp làm một, thế thì năm ba là bạn sẽ thi cùng một team mới hoàn toàn hay sao nhỉ?

Dũng: Năm thứ ba thì mình thi cùng với một bạn từng thi IOI và một bạn Trung Quốc từng thi Putnam.

Long: Thế thì chiến thuật train của bạn như thế nào? Kiểu 2 người nghĩ 1 người code hay là ai nghĩ bài nào thì người đó code?

Dũng: Ở team đi WF lần đầu của mình thì mình có 2 teammate mạnh về toàn diện, nên 2 bạn sẽ cover hầu hết các chủ đề còn mình thì chỉ tập trung vào phần mình giỏi nhất đó là hình học. Cả một năm đó mình chỉ luyện bài hình học thôi. Nhưng cũng tiếc là lúc thi NAC và WF thì mình lại không làm được phần của mình, dù là những kỳ thi khác thì mình đều cân được mọi bài hình học.

Long: Vậy là ai nghĩ thì người đó code luôn đúng không?

Dũng: Đúng rồi.

Dũng: Còn team đi WF lần 2 thì khác, lúc đó thi online vì dịch nên 3 người 3 máy, ở các vòng loại thì bọn mình giải bài độc lập với nhau. Nói chung nếu có teamwork thì bạn từng thi IOI sẽ lo những bài khó nhất, bạn từng thi Putnam sẽ làm những bài Toán, còn mình thì làm những chủ đề còn lại như Data Structures này kia. Đến WF thì chiến thuật đấy của bọn mình lại không hiệu nghiệm nữa vì lúc đấy mọi bài đều rất khó nên bọn mình phải nghĩ ra chiến thuật khác là sau khi giải quyết được xong 1-2 bài dễ thì cả team sẽ tập trung cùng nghĩ bài tiếp theo rồi code. Điều làm mình ngạc nhiên là chiến thuật đấy áp dụng lúc train thì số bài bọn mình làm được tăng lên đáng kể, có đợt train đề WF 2012 team mình còn được giữa bảng xếp hạng. Trước khi thi thì bọn mình cũng hy vọng rất nhiều vào kỳ WF này.

Long: Thế nhưng mà đáng tiếc là vẫn có sự cố nên bạn phải thi WF một mình và được cả hội trường vỗ tay . Thế lúc đó bạn cảm thấy thế nào nhỉ?

Dũng: Trong tiếng Anh có từ bittersweet, kiểu vừa vui vừa buồn. Vui vì lúc đấy team mình được quan tâm hơn, được 2 bạn volunteer xách đồ giúp mình, được các đội đến từ các châu lục cổ vũ rất nhiều, trong đó có các bạn ở Việt Nam.

Dũng: Team mình khi train thì lúc nào cũng nhắm vào top 40, điều đó đối với bọn mình rất giá trị vì rating của mỗi người nhìn vào thì chắc không thể nào lên được tầm đó của ICPC WF rồi. Nhưng khi kết hợp với nhau thì sức mạnh của bọn mình thì được nhân lên rất nhiều, đến tận 3 lần, thường các team khác thì chỉ khoảng 1,5 lần thôi, chắc là vì bộ kĩ năng của bọn mình khá là khác nhau nên khi kết hợp lại thì rất có lợi. Cũng vui nữa là ngay từ đầu bọn mình không quen biết gì nhau cả, do trường chọn team, nhưng sau này lại rất thân với nhau.

Long: Sau WF năm nay rồi thì bạn tốt nghiệp chưa nhỉ?

Dũng: Thực ra mình đã tốt nghiệp trước WF rồi nhưng mà kỳ WF này là một phần của một cái khóa học trong trường mình. Lẽ ra xong WF thì mình mới nhận điểm của khóa học nhưng mà vì mình bị thiệt hại như vậy rồi nên thầy bảo là cứ thoải mái đi thi đi, không lấy kết quả của WF nữa.

[30] APIO: Kỳ thi Olympic Tin học Châu Á -– Thái Bình Dương.

Thế là mình cũng biết là có điểm trước khi thi WF rồi, nhưng mà ngay sau WF thì mình mới chính thức có điểm hết tất cả các môn.

Long: Đó là đại học, vậy dự định trong tương lai của bạn là gì?

Dũng: Hiện tại nếu nói tương lai như 10 năm nữa chẳng hạn thì mình vẫn muốn có một cái bằng Master hoặc PhD. Mình vẫn đang cân nhắc vì các bạn biết đó là một sự đầu tư không nhỏ về thời gian. Ở Mỹ thì từ bằng đại học lên bằng PhD thì sẽ mất khoảng 6 năm, mà trong thời gian đó thì tất nhiên mình có thể đi làm và kiếm được rất nhiều tiền. Cái đấy thì mình vẫn còn đang suy nghĩ chứ hiện tại thì mình chưa học Master, mình vẫn đang đi làm và mình nghĩ là trải nghiệm khi làm việc thì sẽ giúp mình biết được mình thích cái gì và sau này học PhD thì mình sẽ biết được mình nên nghiên cứu mảng nào.

Long: Bạn đang làm trong lĩnh vực nào nhỉ?

Dũng: Mình đang làm một công ty trading, về tài chính. Trong công ty thì mình cũng chỉ là người tính toán những cái công thức rất là phức tạp bằng máy tính thôi. Nhiều cái ban đầu mình thấy rất là đơn giản nhưng khi mà đi vào làm việc thì mới thấy là rất phức tạp. Kiểu nhiều khi có công thức đấy cứ thế mà tính thôi, sau khi mà làm việc thì mình phải chú ý đến độ phức tạp nữa vì cái gì cũng phải tối ưu. Trong trading thì họ đòi hỏi tốc độ xử lý rất nhanh, kiểu nhiều khi O(n) không đủ thì mình phải tìm cách cho nó xuống O(log n) rồi phải cập nhật, mỗi lần cập nhật một cái thông số nào đó thì phải cập nhật luôn kết quả... Nói chung mình thấy kỹ năng trong CP rất là có ích trong công việc của mình.

Dũng: Ngoài ra mình cũng làm interview problemset, tức là mình nghĩ ra bài dành cho interview. Nói chung trong team interview thì sẽ có nhiều CPer nên các bạn làm CP thì cũng có thể quen dạng đề các thứ.

Long: Mình muốn bổ sung thêm phần này thì trong những năm gần đây thì những công ty trading tuyển rất mạnh vào cộng đồng CPer. Vừa rồi ICPC Việt Nam có nhà tài trợ vàng là Optiver, bây giờ các công ty trading cũng tài trợ ICPC rất là nhiều.

Dũng: Đúng rồi, vừa rồi cũng có một công ty trading là Jane Street cũng tài trợ cho ICPC bên mình. Thực sự khi mà đi vào các bài toán thực tế thì đúng là những kỹ năng trong CP mình thấy rất hợp với công việc trong các trading companies.

Long: Vâng, đấy là phần về tương lai. Giờ thì cũng đến phần cuối của phỏng vấn rồi nên như thường lệ mình cũng muốn hỏi vài câu mang tính chia sẻ với các bạn đọc. Dũng là một người đóng góp rất nhiều cho cộng đồng VNOI, không phải ngẫu nhiên mà team VNOI trong lần làm tạp chí này mời Dũng làm người phỏng vấn đầu tiên. Mình nghĩ là bạn là một trong những người có ảnh hưởng rất lớn lên cộng đồng. Đặc biệt trong năm vừa rồi thì mình nhớ là có cái series bvd live rất là được hưởng ứng, thì động lực nào đã khiến bạn làm cái series bvd live chất lượng như vậy?

Dũng: Thực ra động lực cũng chỉ là mình đã nhận được từ VNOI rất nhiều, kiểu mình thấy là dù không phải là nơi giúp mình phát triển nhiều nhất nhưng mà VNOI có nhiều cái đầu tiên của mình, giúp mình tìm được định hướng để phát triển. Nếu như không có VNOI thì mình chắc là sẽ không theo CP. Mình biết Codeforces là qua VNOI, mình biết các site CP khác cũng là qua VNOI. Mình thấy là mình nên có trách nhiệm cho cộng đồng, cống hiến lại cho cộng đồng. Có thể là một bạn nào đấy xem bvd live mà cũng mới học CP thì bạn đó có thể hưởng lợi từ cái series đó và biết đâu bạn lại thành ICPC World Finalist thì sao?

Long: Very inspiring! Rất là đáng quý. Hỏi thêm một tí là bạn có chuẩn bị cái đấy có lâu không?

Dũng: Cũng tùy buổi, có buổi mình chuẩn bị đến 16 tiếng. Mình nhớ cái buổi đấy là về Cây phân đoạn (Segment tree), dù nó là một chủ đề phổ biến trong CP nhưng mình cũng muốn có sự kết nối các buổi với nhau, ví dụ ở đây là liên kết với Chia để trị (Divide and Conquer) vì mình thấy là hiểu được Chia để trị rồi áp dụng vào Segment tree thì có thể giải được các bài khá là sáng tạo. Thế nên là công sức gõ LaTeX rồi vẽ hình các thứ, cũng như là kiểm tra các công thức, cũng gọi là tốn kém nhưng nó đáng thời gian của mình.

Long: Thế thì từ một học sinh học trường không chuyên, sau đấy rồi cũng tự cày lên được thành ICPC World Finalist. Cái thành tích đó cũng rất là đáng quý, ít nhất là ở Việt Nam vì Việt Nam chưa có nhiều người đạt được như vậy. Bạn có lời khuyên gì về cách tự học, cách train và đặc biệt là cách bạn duy trì động lực học được không? Bởi vì ví dụ như bên Chuyên KHTN thì có nhiều bạn học cùng và ai cũng cày căng cực thì điều đấy rất dễ để duy trì động lực học.

Dũng: Thật ra về duy trì động lực thì hồi cấp 3 mình biết là mình kém hơn các bạn rồi nên mình phải cố gắng nhiều hơn. Mình cũng biết là mình không thể cố gắng hơn bên Chuyên KHTN được. Giả sử mỗi ngày mình dành ít nhất 4 tiếng để học

thì có thể là mình cạnh tranh trực tiếp được. Ngoài ra động lực của mình còn là theo dõi những bạn cùng lứa, xem các bạn học thế nào, mình không biết đấy có phải là lời khuyên tốt hay không bởi vì trước tiên bạn phải đánh bại được chính mình đã. Nhưng mà nếu không biết đánh bại chính mình sao thì bạn có thể chuyển sang tìm cách đánh bại người khác .

Dũng: Xem nào, thật ra động lực nữa thì chắc là do mình tham gia các contest thường xuyên, cái rating nó cũng là cái gì đó thúc đẩy mình. Thường thì các bạn nam sẽ quan tâm đến những cái chỉ số để mà cạnh tranh lên. Cái rating thì có thể không phản ánh đúng thực lực của mình nhưng mà nó cũng là một chỉ số để tạo cho mình động lực để cố gắng, nhìn cái rating nó lên cũng thích chứ. Còn hiện tại thì động lực để duy trì CP của mình là mình thấy các kỹ năng nó vẫn còn giúp ích mình trong công việc, mình sẽ tiếp tục luyện tập đến chừng nào mình thấy nó còn giúp ích cho mình. Cũng có nhiều lúc mình tự hỏi là mình học những cái này để làm gì. Tất nhiên là bạn không thể áp dụng 100% những gì mình học nhưng mà vẫn sử dụng một phần nào đấy để áp dụng vào công việc hiện tại. Khi được giao một cái dự án, sếp của mình cũng hay bảo là "Sao mày từng code được bài này thế?", "Sao mày lúc nào cũng từng code mấy cái như này một lần rồi thế?".

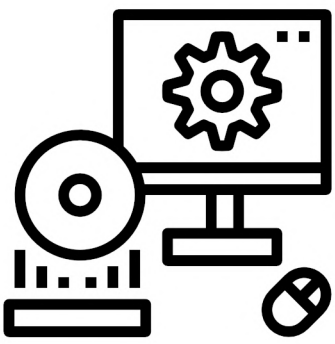
Long: Okay , chắc là câu hỏi cũng chỉ có thế thôi. Các câu chuyện của bạn rất là hay và độc đáo. Để chốt lại thì bạn có điều gì tâm đắc muốn gửi đến bạn đọc không?

Dũng: Có một điều mình nghĩ mình học được qua lập trình đó là tham gia các kì thi Tin học. Nó khác với các kì thi như thi Đại học hay tuyển vào cấp 3, đó là những kì thi mà mình được phép thất bại. Giống như ICPC WF, mình chọn chiến thuật khá là mạo hiểm vì mình biết là mình được phép thất bại. Mình nghĩ các bạn không nên đặt nặng quá, mình không thành công được thì mình thất bại thôi. Cái quan trọng là mình biết nhìn vào những thất bại đó để làm động lực phấn đấu. Mình học được cách biết cười vào những thất bại, những cái ngu của mình trong quá khứ. Lúc đấy việc học và thi đấu Tin học của các bạn nó sẽ thoải mái hơn rất nhiều.

Long: Rất cảm ơn Dũng vì lời nhắn nhủ rất truyền cảm hứng cho các bạn độc giả. Cảm ơn Dũng vì buổi phỏng vấn hôm nay!

Phỏng vấn Trần Quang Lộc

Interviewer: Hoàng Xuân Nhật – Huy chương Bạc IOI 2018



Nhật: Xin chào anh Lộc, mời anh giới thiệu về bản thân ạ.

Lộc: Xin chào các bạn, mình tên là Trần Quang Lộc, sinh năm 1999, nhà ở tỉnh Hòa Bình. Mình vừa tốt nghiệp và đang đi làm tại Pendle. Mình từng là du học sinh Nga, học tại trường ITMO, trường của tourist[31]

Nhật: Anh Lộc có thể kể về quá trình học Tin của mình được không?

Lộc: Mình biết lập trình từ lớp 8, từ môn Tin trên trường là dạy Pascal. Lên cấp 3, mình học trường THPT chuyên Hoàng Văn Thụ, ban đầu mình học chuyên Toán xong đến giữa lớp 11 thì mình chuyển sang học Tin. Trường mình cũng không phải mạnh về môn Tin, nhưng được cái là lúc đó mình cũng rất muốn học Tin và cũng chịu khó tìm tòi các thứ. Mình có đọc cuốn CP[32] 3 nhưng mà chỉ đọc được một nửa thôi, còn lại mình xem đống reference ở dưới rồi tìm nguồn khác đọc.

Lộc: Mình cũng chỉ được thi quốc gia vào lớp 12 và kết quả cũng không được như mình mong đợi, nói một cách trẻ trâu thì gọi là cay cú . Mình nghĩ khả năng mình có thể làm tốt hơn, ở cả quá trình ở cấp 3 nên mình không dừng lại việc học CP mà sẽ tiếp tục theo đuổi trên đại học. Trường ITMO cũng là trường mạnh về CP nên nó cũng giúp ích được cho mình. Mình thì lại không học cùng ngành với tourist, tức là ngành cũng không mạnh về CP lắm nên mình cũng không thi ICPC[33] vào năm nhất. Khi mình thi ICPC bên Nga, mình cũng vừa học vừa ôn thi ICPC. Được cái là team-mate của mình cũng hăng nên mình cũng được tham gia nhiều event, CP training camp. Mình bắt đầu thi ICPC vào năm hai nên mình cũng được thi 3 năm ở Nga.

Nhật: Vậy là từ thất bại ở cấp 3 anh mới quyết tâm try hard trên đại học, điều đó chắc là ai cũng hiểu được. Cơ mà điều mà chắc ai cũng thắc mắc là tại sao anh lại chọn du học Nga?

Lộc: Câu này khá là thú vị! Sau khi mình thi quốc gia thì trường mình có phát động cuộc thi do Trung tâm Văn hóa Khoa học Nga tổ chức để lấy học bổng du học Nga. May mắn thay, mình thi được học bổng 100% nên mình được đi du học mà không phải lo gì về kinh phí cả. Mình được phép chọn trường nên mình chọn luôn ITMO, cũng khá là thuận lợi.

Trần Quang Lộc và trường Đại học ITMO, Nga

Nhật: Vậy cá nhân anh thấy có nên khuyến khích các bạn apply vào học bổng đấy không?

Lộc: Có 2 mặt, phần không nên thì là vì tiếng Nga khó học. Mình phải dành năm đầu khi sang Nga chỉ để học tiếng Nga, tức là mình đi 5 năm. Học xong năm đầu vẫn chưa xong đâu, vào đại học rồi thì vẫn có học phần ngoại ngữ, sinh viên nước ngoài phải học tiếng Nga, còn sinh viên Nga thì phải học tiếng Anh. Cho đến khi mình đến năm cuối rồi thì khả năng nghe tiếng Nga của mình mới gọi là tạm ổn, bởi vì trong lúc học thầy cô thường nói nhanh và nói những từ mình không biết. Nói chung là rào cản ngôn ngữ sẽ lớn hơn rất nhiều so với đi du học ở những nơi chỉ cần tiếng Anh.

Nhật: Vậy là anh học hoàn toàn bằng tiếng Nga?

Lộc: Đúng vậy, nhưng mà để mình kể cho cái này vui này! Dù mình phải nghe giảng bằng tiếng Nga nhưng mà 2 năm đầu thì mình được xin thi cử bằng tiếng Anh. Lên năm 3, trình độ tiếng Nga của mình mới lên hẳn. Với thêm một lí do nữa cho điều đó là vì mình thi ICPC . Đơn giản là vì mình phải giao tiếp với teammate, mình đọc đề tiếng Anh rồi mình phải giải thích đề bằng tiếng Nga. Mình cố phải dùng tiếng Nga chứ không muốn sử dụng tiếng Anh với teammate cho nó tự nhiên hơn. Nó tạo thành một cái vòng lặp cho mình để mà mình tiến bộ.

Lộc: Thêm vài lý do không nên đi Nga nữa là COVID và đang có xung đột. Bây giờ dịch cũng đỡ rồi nhưng nếu xảy ra sự kiện tương tự như thế làm bạn phải ở lại bên Nga, mà Nga thì lại rất xa nên chắc chắn là người nhà cũng lo, mình cũng buồn. Cái xung đột bên Nga cũng ảnh hưởng ít nhiều đến

[31] tourist: tên thật là Gennady Korotkevich - một lập trình viên người Belarus. Anh có 6 Huy chương vàng IOI (2007 -– 2012) và 2 lần vô địch thế giới tại kỳ thi ICPC (năm 2013 và 2015).

[32] CP: Competitive Programming - Lập trình thi đấu

[33] ICPC: International Collegiate Programming Contest - Cuộc thi Lập trình Quốc tế lâu đời và danh giá nhất dành cho sinh viên các trường đại học và cao đẳng trên toàn cầu.

cá nhân bạn.

Lộc: Còn về phần nên, nếu bạn là người thích đi đây đi đó thì sẽ thích Nga, vì Nga có rất nhiều cảnh đẹp, cũng là một trải nghiệm khá vui. Thêm nữa là nếu bạn may mắn thì sẽ được dự các sự kiện hay ho bên Nga, giống như mình được đi training camp rất vui. Mình cũng thấy là Regional ở Nga tổ chức rất là hoành tráng. À, nếu bạn chọn ITMO thì cũng có nhiều người Việt nên bạn cũng không cảm thấy cô đơn khi sang đây, không thiếu người để nói tiếng mẹ đẻ cùng . Trong ký túc xá mình có đến tầm 200 người Việt!

Nhật: Nói chung vừa nãy anh nói thì em mới nhận ra là việc thi ICPC với teammate Nga nó khó hơn hẳn thi với teammate nói tiếng Anh. Cho nên em thấy điều đó khiến thành tích của anh nó ngầu hơn em nghĩ! Lúc nãy anh cũng có nói là anh và tourist học khác ngành nên chắc mọi người cũng thắc mắc rằng học CP thì sau này ra làm gì? Chẳng hạn như tourist giờ làm gì, anh làm gì, và 2 người khác ngành thì có khác công việc không?

Lộc: tourist hiện tại theo mình biết là tiếp tục làm nghiên cứu sinh tại trường. Hai năm cuối mình thi thì tourist là coach ICPC trường mình luôn. Theo mình biết, lĩnh vực nghiên cứu của tourist là tin sinh học, mình cũng thấy là học CP có lợi cho lĩnh vực đấy, kiểu xài các thuật toán xử lý chuỗi trong gen ấy. Còn phần mình, hiện tại mình đang làm fullstack, những thứ mình code thì nó về thiết kế phần mềm hơn là sử dụng thuật toán. Tuy nhiên các bài toán mình giải quyết vẫn là tốc độ và độ ổn định. Nếu bạn có kinh nghiệm làm CP thì phần tốc độ cũng sẽ thuận lợi cho bạn.

Nhật: Vậy sau khi sang Nga, anh thấy mình đã đạt được mục tiêu anh muốn trong CP chưa?

Lộc: Tất nhiên mục tiêu cao nhất của mình là vào được WF[34] , và mình đã không làm được. Nhưng mình cũng được thi Regional, và cái Regional của mình (NERC[35]) cũng được đánh giá là khó. Thêm một điều nữa là mình được gặp tourist này! Mình cũng thấy đó là điều tuyệt vời mà không phải ai cũng được.

Trần Quang Lộc với đội Unexpected Value trong kì thi ICPC NERC 2021

Nhật: Không chỉ tourist mà anh còn được gặp founder Codeforces[36] Mike Mirzayanov[37] đúng không?

Lộc: Đúng rồi! Mike thật ra làm ở tỉnh khác, sau đó thì mới chuyển sang công tác ở ITMO. Mike thì cũng làm coach khá lâu rồi, nhưng mà giờ ông ý cũng không thích làm nữa mà lại thích làm về dev hơn, còn tourist thì vẫn dạy algo.

Nhật: Nhân lúc nói về Mike thì anh có thể chia sẻ luôn câu chuyện anh làm coordinator cho Codeforces được không? Anh cũng là người Việt Nam đầu tiên có title đó luôn đúng không?

Lộc: Đúng rồi, thật ra cũng đơn giản! Vào một ngày bình thường, mình thấy tin nhắn trên Codeforces bằng tiếng Nga huy động một số người làm coordinator, ai muốn làm thì liên hệ với Mike hoặc với KAN[38] . Lúc đấy mình lên đỏ rồi! Trước đó mình cũng không có kinh nghiệm ra đề với coordinate contest lắm, chính cái round Codeforces mình làm là một bài kiểm tra và bài hướng dẫn luôn!

Nhật: Vậy anh có thể chia sẻ thêm rằng làm coordinator có vui không? Anh có recommend các bạn khác làm coordinator giống anh không?

Lộc: Về làm coordinator, các bạn nếu thấy mình khủng thì có thể nhắn với KAN. Bởi vì errorgorn[39] bảo với mình là nó thích nên nó nhắn

[34]WF: World Finals – Cuộc thi ICPC cuối cùng trong một mùa giải bao gồm những đội xuất sắc nhất của các trường đại học trên thế giới vượt qua vòng loại vùng (Regional contest)

[35]NERC: Northern Eurasia Regional Contest – Kỳ thi ICPC vòng loại khu vực Bắc Á-Âu.

[36]Codeforces: một trang web tổ chức các cuộc thi lập trình thi đấu với các dạng đề, bài tập đa dạng.

[37]Mike Mirzayanov: một lập trình viên người Nga, học và tốt nghiệp tại trường ITMO. Thường được biết đến là người tạo ra nền tảng chấm bài trực tuyến Codeforces.

[38]KAN: tên thật là Nikolay Kalinin - một lập trình viên người Nga. Anh có 2 Huy chương vàng IOI (2013 – 2014) và 1 lần vô địch thế giới tại kỳ thi ICPC (năm 2020). Thường được biết đến là người phụ trách hỗ trợ tổ chức kỳ thi trên nền tảng chấm bài trực tuyến Codeforces. Xem thêm tại https://cphof.org/profile/topcoder:KalininN.

[39]errorgorn: tên thật là Ashley Khoo - một lập trình viên người Singapore. Anh có 3 lần tham gia kì thi IOI, trong đó có 2 Huy chương vàng IOI (2021 - 2022) và 1 Huy chương bạc IOI (2020). Xem thêm tại https://cphof.org/profile/codeforces:errorgorn.

KAN xin vô làm thôi. Các bạn nên hiểu là việc của coordinator không phải là làm round mà là đảm bảo chất lượng của round, về đề lẫn về test. Một là đề phải đọc vào dễ hiểu, đầy đủ thông tin, hai là test phải mạnh. Bạn sẽ phải hướng dẫn các bạn viết problem, rồi phải hướng dẫn sinh test như thế nào, nói chung thì cái này cũng có format. Trước khi bài được cho vào round thì coordinator phải duyệt bài nữa, có thể từ chối thoải mái . Cái hay của việc đó là bạn được biết những ý tưởng hay từ những bài được đề xuất. Hầu hết bài bây giờ không phải là bài tự sáng tác nữa, mình cũng phải đọc, ngẫm, xem solution của tác giả có hợp lệ không. Theo mình đó là phần thú vị khi làm coordinator .

Lộc: Nói chung là phải đảm bảo hết các giai đoạn của quá trình làm contest, chỗ nào không ổn thì phải sửa ngay. Ví dụ test xong round mà thấy bài này không ổn thì phải thay bằng bài khác, nhiều khi công đoạn đó phải lặp lại nhiều lần khá lâu.

Nhật: Khá là thú vị, và cũng công nhận là từ khi sếp Lộc coordinate cho contest của VNOI thì chất lượng của nó cũng lên hẳn, ví dụ như các contest Bedao gần đây. Vậy thì anh có thể chia sẻ quá trình anh vào VNOI được không?

Lộc: Mình vào VNOI từ hè 2021, khi mà VNOI mở đợt TNV gen 1 thì mình có apply. Một thời gian sau khi mình bắt đầu làm TNV, sếp Lomk cũng mời mình thành lead của TNV. Sau một thời gian nữa, chắc là công việc mình làm tốt nên mình lại được mời làm admin VNOI .

Nhật: Vậy cấp 3 anh có biết đến VNOI không?

Lộc: Phải vào đội tuyển rồi thì mình mới biết, trước đó mình gần như chả biết gì cả! Lúc đấy mình join group VNOI trên Facebook, rồi cũng hay trả lời các bạn hỏi bài này nọ. Hình như lúc đấy là group VNOI trên Facebook đang xôm và may mắn là mình cũng active nhất vào lúc đấy . Lên đại học, mình cũng không active trên Facebook nữa, group VNOI cũng ít hoạt động dần. Nhưng VNOI lại có thêm server trên Discord và mình lại active trên Discord . Kiểu có bạn nào ping mình thì mình lên đấy trả lời giúp hỏi bài các kiểu. Mình cũng biết đợt tuyển TNV là qua VNOI Discord server.

Nhật: Vậy giờ anh còn active trên Discord nữa không?

Lộc: Thỉnh thoảng có bạn hỏi bài trên đấy thì mình vẫn vào trả lời, bây giờ mình cũng tập trung vào công việc hơn nên chắc là sẽ không active bằng lúc trước. Phần nữa là mình cũng không tập trung vào CP nhiều nữa.

Nhật: Anh có thể chia sẻ thêm về những câu chuyện trên đại học của anh được không?

Lộc: Mình ở ký túc xá. Ở bên Nga thì round Codeforces diễn ra vào 5h chiều, mình cũng hay đi học về rồi nếu có contest thì thi luôn, kiểu vừa ăn tối vừa thi . Cũng mấy lúc chờ system testing luôn, nhưng mà không được để nhà tắm đóng cửa .

Nhật: Nghe thú vị thế, anh có thể kể thêm về chi tiết đó để các bạn biết về trải nghiệm sử dụng nhà tắm công cộng không? Ở Việt Nam thì không có cái này.

Lộc: Ừ đúng! Cái nhà tắm thì dành cho năm nhất đến năm ba xài chung, năm tư thì xài riêng cái khác, mở từ khoảng 8h sáng đến 9h tối. Nhiều khi mình thi CF xong mải chat mải hóng system testing thì lại nhận ra nhà tắm sắp đóng cửa, thế là phải chạy đi ngay . Năm tư thì trường xây thêm nhà tắm nên được xài riêng, lúc đó cũng đang COVID nên là cũng cần thiết. Có mấy lúc mình vào nhà tắm thì có mỗi một mình trong đấy (bình thường thì đông lắm nhé).

Lộc: Ngoài nhà tắm ra thì còn có nhà bếp nữa. Các bạn cũng được vào đó tự nấu đồ của mình các thứ các thứ, cũng vui. Giờ thì nó hơi xuống cấp tí rồi . Mình cũng thấy là các bạn Nga ít khi tự nấu, mà người Nga cũng ăn tối đơn giản lắm, bữa trưa mới gọi là đầy đủ.

Nhật: Nếu vậy thì khi anh còn đi học đại học, thời gian biểu một ngày của anh trông như thế nào?

Lộc: Câu hỏi thú vị, một ngày ở trường thì có khoảng 7 tiết. Tiết đầu bắt đầu từ 8h sáng, nhưng mà các bạn nhớ là dậy vào 8h ở Nga nó khó hơn dậy ở VN nhé ;(. Tiết cuối cùng kết thúc vào 8h tối. Mình thường không phải học đến 8h tối vì mình chỉ học khoảng 5 tiết là nhiều nhất. Mỗi tiết của mình kéo dài 1 tiếng 30 phút. À, mà trường mình cũng có nhiều cơ sở nên có mấy lúc mình cũng phải di chuyển giữa các cơ sở ấy.

Nhật: Ngoài đi học ra thì anh còn làm gì nữa?

Lộc: Tất nhiên là phải kể đến thời gian mình di chuyển từ ký túc xá lên trường và ngược lại . Ở Nga thì không có tàu điện hay xe buýt nên mình phải đi bộ tầm 40 phút cho một chuyến. Được cái vào năm tư thì có dịch vụ thuê xe điện nên mình được di chuyển tiện hơn, cái này có app hẳn hoi nhé. Trường mình cũng liên kết với các công ty nên mỗi sinh viên cũng được 1 vé miễn phí đi trong ngày . Trong 1 tuần thì chỉ được 5 ngày thôi cho nên nếu muốn đi vào T7 thì vẫn phải đi bộ ;(

Nhật: Thế anh có phải làm bài tập về nhà nhiều không?

Lộc: Cũng có, đặc biệt là các lớp lab này. Nhiều môn làm thì rất nhanh nhưng mà phải làm báo cáo còn lâu hơn thời gian làm nhiều. Có một môn là ông thầy bắt mình AC khoảng 20 bài trên acm.timus.ru, tất nhiên là thời gian để làm thế thì rất nhanh với mình . Nhưng phần mệt nhất là phải vẽ sơ đồ khối cách giải các bài đấy, phần đầu lâu cực.

Nhật: Lúc trước khi qua Nga thì Codeforces anh màu gì nhỉ?

Lộc: Lúc mình thi xong lớp 12 thì mình ở tím.

Nhật: Cái này em nghĩ nhiều bạn cũng quan tâm nhiều. Anh lên đại học rồi thì thời gian cho CP cũng giảm xuống so với cấp 3 nhưng mà anh lại tiếp tục cày lên đỏ CF. Vậy anh đã làm được điều đó như thế nào nhỉ?

Lộc: Chắc là do mình vẫn làm contest thường xuyên nên mình vẫn duy trì được trình độ, với lại mình cũng hay luyện bằng bot TLE trên server của VNOI, mỗi ngày mình gitgud 1 lần. Mình thấy cái đó cũng hay vì là mỗi lần gitgud rồi AC, dù mình tự giải hay là bí đọc lời giải rồi code lại thì nó cũng tính điểm cho mình, từ đó cũng hình thành một cái quá trình để mình theo dõi. Đó cũng là một động lực để mình tiếp tục luyện.

Lộc: Thực ra thì quá trình luyện của mình cũng lâu chứ, mãi đến kỳ 1 của năm tư mình mới lên được đỏ. Thực ra lúc làm contest, mình cũng trên tinh thần là thi cho vui, vì vui nên lần sau mình lại thi tiếp. Nhiều bạn có thể không nghĩ giống mình nên các bạn bỏ contest khi lên đại học. Ngoài ra làm contest nhiều cũng giúp bạn luyện được tư duy, đó cũng là lợi thế cho các việc khác. Dù gì thì các bạn cũng nên coi CP là một môn thể thao chứ không phải là cái gì giúp bạn tăng lương hay gì đấy . Thích thì mình mới làm!

Nhật: Một câu hỏi không liên quan lắm nhưng sao anh lại đặt handle là darkkcyan?

Lộc: Vì mình thích màu cyan mà đậm, cho nên mình lấy darkkcyan . À thực ra vì sao nó lại có 2 chữ k? Lúc trước vì lí do nào đấy mà tên darkcyan bị chiếm trước nên mình đặt thế . Mình cũng có đưa câu chuyện này vào một bài trong VNOI Cup 2022, các bạn có thể xem lại .

Nhật: Vậy thì trên đại học anh có kiểu muốn đánh bại một ai đó không, trong CP ấy? Em thấy nhiều bạn hay tập trung vào ganh đua hơn nhiều.

Lộc: Thực ra cái này thì mình tập trung vào bản thân hơn. Mình không cho phép cái đấy làm động lực để mình phấn đấu vì nó không hấp dẫn với mình lắm nên mình cũng không để ý nhiều đến ai khác. Thực ra thì lúc nào cũng có những cái tên ở top đầu như tourist, um_nik thì ai cũng biết khủng và mình chắc là không thể đánh bại nổi rồi. Thời học ở cấp 3, đội tuyển mình cũng nhàn, bao nhiêu bạn học đội tuyển thì bấy nhiêu suất đi thi quốc gia, cũng không ganh đua gì lắm. Mình thấy cái động lực ganh đua với người khác thì một là đánh bại được rồi mình lại hết động lực, hai là mãi vẫn không đánh bại được nên cũng chán, chán thì lại bỏ. Nói chung cái động lực đó mình thấy nó không bền vững.

Nhật: Có một cái em khá ngưỡng mộ anh đó là ngoài CP anh còn biết code nhiều thứ khác nữa, như cái trang blog của anh nhìn cũng khá đẹp. Anh có thể chia sẻ thêm về quá trình tự học lập trình ngoài CP không?

Lộc: Thực ra hồi đó mình học Javascript trước khi học C++, lúc đó Javascript còn chưa có class, nhiều lúc mình cũng thấy Javascript rối rắm. Nhưng mà mình kiểu thấy ngôn ngữ này hay thì mình cứ học cho vui, thế là mình biết những cái hay cái dở của từng ngôn ngữ, như Java, Kotlin, Lua, Python này. À nhắc đến Python thì mình cũng hay xài Python để tính toán, mình cũng hay xài Jupyter notebook để làm thống kê thay Excel, một công cụ rất là mạnh. Ngoài ra còn có LaTeX, mình cứ làm báo cáo bằng LaTeX hết cho sướng . Mình cũng tìm ra cách để sinh code LaTeX bằng Python . Mình thấy các bạn chịu khó tìm hiểu thì chắc chắn sẽ tìm ra được nhiều thứ hay ho để làm. Mình đánh giá là làm kiểu của mình - kiểu tìm cách tự động hóa thì thời gian làm cũng không chênh lệch với các bạn làm thủ công. Nhưng có một cái lợi thế của cách mình đó là sửa rất nhanh , còn các bạn thì nhiều khi phải làm lại từ đầu khi muốn sửa, rất là mất thời gian.

Lộc: Về việc tìm hiểu về công nghệ thì mình cũng thấy việc đó không khó bởi vì documentation của nó thì đã có trên mạng rồi và mình chỉ việc đọc rồi thực hành thôi. Cái các bạn cần chú ý là phải có tiếng Anh, bởi vì hầu hết mọi thứ trên mạng đều bằng English. Nói chung, mình là người chịu khó đi đường dài, mình không thích chọn đường tắt. Mình muốn chọn đường dài để hiểu tường tận mọi thứ hơn. Cứ thấy mình đi được dài thì mình sẽ đi, các bạn chắc chắn sẽ học được nhiều thứ trong cuộc sống khi làm vậy.

Nhật: Cuối cùng thì anh có điều gì muốn nhắn nhủ với các bạn độc giả của tạp chí VNOI lần này không?

Lộc: Nếu các bạn thấy con đường của mình thú vị thì các bạn có thể thử . Mình làm mọi thứ khá là đơn giản, thứ duy nhất mình thấy mình phải đánh đổi là thời gian. Nói chung thì như mình nói

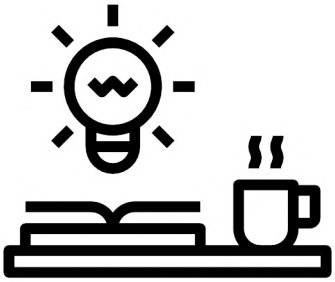
đấy, các bạn nên chịu khó tìm hiểu các thứ, chắc chắn các bạn cũng tìm được nhiều thứ hay ho, và trong quá trình lâu dài thì chắc chắn các bạn sẽ học được nhiều điều mà các bạn đi đường ngắn sẽ không có được.

Nhật: Rất cảm ơn những chia sẻ của anh Lộc.

Lộc: Cảm ơn các bạn rất nhiều!

Khử Gauss-Jordan

Nguyễn Đức Anh
11 Tin, Trường THPT Chuyên Hà Nội - Amsterdam



Giới thiệu

Cho một hệ n phương trình đại số tuyến tính (system of linear algebraic equations - SLAE) vớim ẩn. Ta được yêu cầu giải hệ phương trình đó (tức là xác định xem nó có vô nghiệm, chính xác một nghiệm hay vô số nghiệm) và trong trường hợp hệ có ít nhất một nghiệm, hãy đưa ra một nghiệm bất kì của hệ đó.

Nói tóm lại, ta được yêu cầu giải hệ phường trình sau:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + &\quad \dots \quad + a_{1m}x_m &&= b_1 \\ a_{21}x_1 + a_{22}x_2 + &\quad \dots \quad + a_{2m}x_m &&= b_2 \\ &\quad \vdots \\ a_{n1}x_1 + a_{n2}x_2 + &\quad \dots \quad + a_{nm}x_m &&= b_n \end{aligned}$$

trong đó a_{ij} $(1 \leqslant i \leqslant n$ và $1 \leqslant j \leqslant m)$ và b_i ($1 \leqslant i \leqslant n)$ là các hệ số đã biết còn x_i $(1 \leqslant i \leqslant m)$ là các ẩn.

Ngoài ra còn có cách biểu diễn hệ bằng ma trận như sau:

$$Ax = b$$

trong đó A là ma trận kích thước $n \times m$ chứa các hệ số a_{ij} và b là vector độ dài n chứa các hệ số b_i.

Đặc biệt, phương pháp này còn có thể áp dụng trong trường hợp các phương trình có kết quả lấy dư cho số nguyên dương p bất kì:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \dots + a_{1m}x_m \equiv b_1 \pmod p \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2m}x_m \equiv b_2 \pmod p \\ \quad \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nm}x_m \equiv b_n \pmod p \end{cases}$$

Thuật toán

Phương pháp khử Gauss-Jordan dùng cách lần lượt khử các ẩn để đưa hệ phương trình đã cho về một dạng ma trận rồi giải hệ phương trình. Cụ thể hơn, ta sẽ dùng phương trình thứ nhất để khử ẩn x_1 trong $n-1$ phương trình còn lại, tức là biến tất cả các hệ số a_{i1} $(2 \leqslant i \leqslant n)$ thành 0 mà hệ phương trình mới vẫn tương đương với hệ cũ. Tương tự, ta tiếp tục dùng phương trình thứ hai để khử ẩn x_2 trong $n-1$ phương trình còn lại, ...

Đầu tiên ta sẽ định nghĩa 2 phép biến đổi sau:

- Nhân phương trình với số thực r khác 0: nhân tất cả các hệ số của phương trình với r.
- Cộng hai phương trình trong hệ với nhau: Cộng lần lượt các hệ số của phương trình này vào các hệ số của phương trình kia.

Ta dễ dàng thấy sau khi thực hiện các phép biến đổi trên thì hệ mới vẫn tương đương với hệ cũ.

Để thực hiện phép khử Gauss, ta sẽ lần lượt dùng các phương trình khác nhau để khử ẩn x_i $(1 \leqslant i \leqslant min(n, m))$:

- Tìm phương trình e chưa được sử dụng mà có $a_{ei} \neq 0$:
  - Nếu tìm được: Chia tất cả các hệ số của phương trình e cho a_{ei}, đánh dấu phương trình e đã được sử dụng và lưu ẩn x_i được phương trình e khử.
  - Không tìm được: Khi đó x_i sẽ là ẩn tự do và ta chuyển sang khử ẩn x_{i+1} (nói cách khác thì x_i sẽ là một giá trị bất kì).
- Với $n-1$ phương trình r còn lại $(1 \leqslant r \leqslant n$ và $r \neq e)$, ta sẽ cộng phương trình r với phương trình e nhân $-a_{ri}$. Khi đó, tất cả các hệ số a_{ri} là 0 còn a_{ei} là 1.

Sau khi khử xong ta sẽ bắt đầu tìm nghiệm:

- Nếu $n < m$ thì tất cả các ẩn x_i $(n < i \leqslant m)$ sẽ là ẩn tự do.
- Ta sẽ cho luôn các ẩn tự do bằng 0.
- Với các ẩn x_i không phải ẩn tự do, gọi e là phương trình đã dùng để khử x_i. Khi đó $a_{ei} = 1$ và tất cả các hệ số a_{ej} còn lại sẽ bằng 0 hoặc x_j là ẩn tự do và đã được đặt bằng 0. Từ đó suy ra $x_i = b_e$.
- Sau đó ta sẽ thử lại hệ phương trình:
  - Không thỏa mãn: hệ vô nghiệm.
  - Thỏa mãn: hệ có ít nhất 1 ẩn tự do thì hệ có vô số nghiệm còn không thì hệ sẽ có 1 nghiệm duy nhất.

Ví dụ:

- Ta sẽ sử dụng ma trận kích thước $n \times (m+1)$ để biểu diễn hệ phương trình. Mỗi dòng của ma trận là 1 phương trình, mỗi phương trình có m số đầu tiên là hệ số a_{ij} và số cuối cùng là b_i.
- Cho hệ phương trình sau:

$$\begin{cases} x + 5y + 3z - 4t = 19 & (1) \\ 3x + 16y + 7z - 9t = 42 & (2) \\ -2x - 8y - 9z + 3t = -61 & (3) \end{cases} \qquad \begin{pmatrix} 1 & 5 & 3 & -4 & 19 \\ 3 & 16 & 7 & -9 & 42 \\ -2 & -8 & -9 & 3 & -61 \end{pmatrix}$$

- Dùng phương trình (1) để khử ẩn x:
  - Trừ phương trình (2) đi 3 lần phương trình (1).
  - Cộng phương trình (2) với 2 lần phương trình (1). $\Longrightarrow$ Hệ số của ẩn x ở phương trình (2) và (3) sẽ trở thành 0.

$$\begin{cases} x+5y+3z-4t=19 & (1) \\ y-2z+3t=-15 & (2) \\ 2y-3z-5t=-23 & (3) \end{cases} \qquad \begin{pmatrix} 1 & 5 & 3 & -4 & 19 \\ 0 & 1 & -2 & 3 & -15 \\ 0 & 2 & -3 & -5 & -23 \end{pmatrix}$$

- Dùng phương trình (2) để khử ẩn y:
  - Trừ phương trình (1) đi 5 lần phương trình (2).
  - Trừ phương trình (3) đi 2 lần phương trình (2). $\Longrightarrow$ Hệ số của ẩn y ở phương trình (1) và (3) sẽ trở thành 0.

$$\begin{cases} x+13z-19t=94 & (1) \\ y-2z+3t=-15 & (2) \\ z-11t=7 & (3) \end{cases} \qquad \begin{pmatrix} 1 & 0 & 13 & -19 & 94 \\ 0 & 1 & -2 & 3 & -15 \\ 0 & 0 & 1 & -11 & 7 \end{pmatrix}$$

- Dùng phương trình (3) để khử ẩn z:
  - Trừ phương trình (1) đi 13 lần phương trình (3).
  - Cộng phương trình (2) với 2 lần phương trình (3). $\Longrightarrow$ Hệ số của ẩn z ở phương trình (1) và (3) sẽ trở thành 0.

$$\begin{cases} x+124t=3 & (1) \\ y-19t=-1 & (2) \\ z-11t=7 & (3) \end{cases} \qquad \begin{pmatrix} 1 & 0 & 0 & 124 & 3 \\ 0 & 1 & 0 & -19 & -1 \\ 0 & 0 & 1 & -11 & 7 \end{pmatrix}$$

- Cuối cùng, vì t là ẩn tự do nên ta sẽ đặt $t=0$. $\Longrightarrow$ Ta có một nghiệm sau:

$$\begin{cases} x=3 \\ y=-1 \\ z=7 \\ t=0 \end{cases}$$

- Ngoài ra, vì có 1 ẩn tự do là t nên hệ có vô số nghiệm. Một nghiệm khác thỏa mãn là:

$$\begin{cases} x=-121 \\ y=18 \\ z=18 \\ t=1 \end{cases}$$

Lưu ý

- Khi chọn phương trình e có $a_{ei} \neq 0$, ta nên chọn e có $|a_{ei}|$ lớn nhất để các hệ số của phương trình e trở nên nhỏ hơn và tránh tràn số.

Cài đặt

- Sử dụng `vector<vector<double> > a` để lưu hệ phương trình.
- Dùng swap để tất cả các phương trình được sử dụng rồi đều được cho lên đầu của a. $\Rightarrow$ Dùng 2 còn trỏ row và col cho biết chỉ còn những phương trình từ row đến $n-1$ chưa được sử dụng và đang khử đến ẩn x_{col}.
- `where[i]` để lưu vị trí phương trình khử ẩn x_i.
- Hàm `int gauss(vector<vector<double> > a, vector<double> &ans)` để giải hệ. Hàm sẽ trả về:
  - 0: hệ vô nghiệm.
  - 1: hệ có một nghiệm duy nhất.
  - 2: hệ có vô số nghiệm.

  Trong trường hợp hệ có ít nhất một nghiệm thì ans sẽ lưu 1 nghiệm của hệ phương trình.

```
const double eps = 1e-9;

int gauss(vector<vector<double>> a,
          vector<double> &ans) {
  int n = (int)a.size(); // số phương trình
  int m = (int)a[0].size() - 1; // số ẩn

  vector<int> where(m, -1);
  for (int col = 0, row = 0; col < m && row < n;
       col++) {
    int e = row;
    for (int i = row + 1; i < n; i++) {
      if (abs(a[i][col]) >
          abs(a[e][col])) { // chọn phương trình e
        // có abs(a[e][col]) lớn nhất
        e = i;
      }
    }

    if (abs(a[e][col]) <
        eps) // tất cả các hệ số a[e][col] đều
             // bằng 0
      // => ẩn x[col] là ẩn tự do
      continue; // chuyển sang khử x[col + 1]

    swap(a[e],
         a[row]); // chuyển phương trình e lên đầu

    where[col] =
      row; // ẩn x[col] được khử bởi phương trình
           // đang ở vị trí row

    /* chuyển hệ số a[row][col] về 1 */
    double tmp = a[row][col];
    for (int i = 0; i <= m; i++) {
      a[row][i] /= tmp;
    }

    /* khử hệ số a[i][col] ở toàn bộ n - 1 phương
     * trình còn lại */
    for (int i = 0; i < n; i++) {
      if (i != row) {
        double c = a[i][col];
        for (int j = col; j <= m;
             j++) { // chỉ cần for từ col tại vì
          // a[row][p] (p < col) đã bằng 0
          a[i][j] -= a[row][j] * c;
        }
      }
```

```
    }

    row++; // phương trình ở vị trí row đã được sử
           // dụng
  }

  ans.assign(m, 0); // gán tất cả các ẩn ban đầu 0
  for (int i = 0; i < m; i++) {
    if (where[i] != -1) {
      ans[i] =
        a[where[i]]
         [m]; // nếu x[i] không phải ẩn tự do thì
      // x[i] = b[where[i]]
    }
  }

  /* Thử lại */
  for (int i = 0; i < n; i++) {
    double sum = 0;
    for (int j = 0; j < m; j++) {
      sum += ans[j] * a[i][j];
    }
    if (abs(sum - a[i][m]) > eps) // sum != b[i]
      return 0;
  }

  for (int i = 0; i < m; i++)
    if (where[i] ==
        -1) // nếu có ít nhất 1 ẩn tự do thì
      // hệ có vô số nghiệm
      return 2;
  return 1;
}
```

Sau khi cài đặt xong, bạn đọc có thể nộp thử ở đây [40] .

Độ phức tạp

Phần khử các ẩn:

Ta phải thực hiện khử các ẩn $min(n, m)$ lần:

- Ta phải tìm phương trình e chưa sử dụng có $|a_{ei}|$ lớn nhất mất $\mathcal{O}(n)$.
- Khử ẩn x_i ở tất cả các phương trình khác mất $\mathcal{O}(nm)$.

Phần tìm nghiệm:

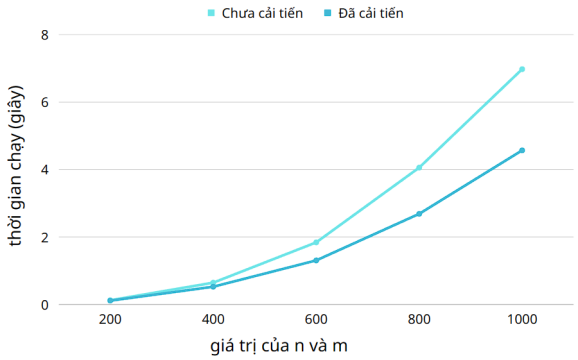
Ta mất $\mathcal{O}(m)$ để tìm tất cả các nghiệm và mất $\mathcal{O}(nm)$ để thử lại.

$\Longrightarrow$ Độ phức tạp cuối cùng sẽ là $\mathcal{O}(min(n, m)nm)$. Trong trường hợp $n = m$ thì độ phức tạp chỉ đơn giản là $\mathcal{O}(n^3)$.

Độ phức tạp bộ nhớ: $\mathcal{O}(nm)$.

Cải tiến tốc độ

Với cách cài đặt ở trên, mỗi lần khử ẩn x_i ta lại phải thực hiện phép cộng với tất cả $n - 1$ phương trình còn lại, kể cả các phương trình đã sử dụng. Vì thế ta có thể bỏ qua không cộng các phương trình đã sử dụng và giảm số lần cộng đi một nửa. Nếu ta bỏ qua các phương trình đã sử dụng thì một phương trình vẫn có thể có nhiều hệ số a_{ij} khác 0 (với các $j > n$ ta tạm coi a_{ij} là 0 bởi vì x_j là hệ số tự do và đã được cho là 0). Tuy nhiên, ở phương trình e_n để khử ẩn x_n chỉ có $a_{e_n n} \neq 0$, phương trình e_{n-1} chỉ có $a_{e_{n-1}n-1} \neq 0$ và $a_{e_{n-1}n} \neq 0$, ... (Hay nói cách khác, thay vì ma trận A trở thành ma trận đơn vị thì nó sẽ trở thành hình tam giác). Từ đó, ta có thể dùng phương trình e_n để tính x_n rồi thay vào phương trình e_{n-1} để tính x_{n-1}, ... Vì thế, việc tìm nghiệm chỉ tốn $\mathcal{O}(nm)$ còn việc khử ẩn thì đã giảm được một nửa số lần phải cộng 2 phương trình. $\Longrightarrow$ Tốc độ chạy gần như nhanh gấp đôi.



Giải các hệ có modulo

Việc giải các hệ phương trình có modulo sẽ hơi khác ở phần khử ẩn vì các hệ số đều là số nguyên. Ta sẽ không chia để cho hệ số a_{ei} về 1 nữa mà nhân phương trình r với a_{ei} rồi trừ đi phương trình e nhân a_{ri}. Từ đó hệ số a_{ri} sẽ vẫn về 0.

```
for (int r = 0; r < n; r++) {
  if (r != e) {
    S[r] = S[r] * S[e].a[i] + S[e] * (-S[r].a[i]);
  }
}
```

Tuy nhiên, với trường hợp modulo 2 ta có thể sử dụng bitset để tăng tốc độ xử lý. Ở cách cài đặt dưới đây, ta sẽ chỉ dùng `vector<bitset<maxm> >` để biểu diễn cả hệ phương trình (bit thứ m của phương trình i sẽ là b_i) và `ans` sẽ là `bitset<maxm>`.

[40] https://hnoj.edu.vn/problem/khu_gauss

```
int gauss(vector<bitset<maxm>> a,
          bitset<maxm> &ans) {
  int n = a.size();
  int m = maxm - 1;
  vector<int> where(m, -1);
  for (int col = 0, row = 0; col < m && row < n;
       col++) {
    for (int i = row; i < n; i++) {
      if (a[i][col]) {
        swap(a[i], a[row]); // a[e][col] lớn nhất
                            // có thể chỉ là 1
        break;
      }
    }

    if (!a[row][col]) continue;

    where[col] = row;

    for (int i = 0; i < n; i++) {
      if (i != row && a[i][col]) {
        a[i] ^= a[row]; // Việc khử ẩn chỉ cần
                        // dùng phép XOR là được
        // nên tốc độ xử lí nhanh hơn rất nhiều
      }
    }
    row++;
  }
  // Phần tìm nghiệm tương tự như trên
}
```

Ứng dụng

Ngoài ứng dụng cho việc giải hệ phương trình, khử gauss còn có thể áp dụng vào tìm số nghiệm của phương trình có modulo, tìm số ẩn tự do, tìm ma trận nghịch đảo, ... Tuy nhiên, ở đây chỉ có những ứng dụng đơn giản của khử gauss.

Ví dụ 1 – FINDGRAPH - HNOJ 41

Đề bài

Cho đồ thị vô hướng và số nguyên tố p, mỗi đỉnh có trọng số x_i $(0 \leqslant x_i < p)$. Mỗi đỉnh u sẽ có một số nguyên b_u $(0 \leqslant b_u < p)$ thỏa mãn $b_u = \Sigma x_v (mod\ p)$ với tất cả các đỉnh v có cạnh nối trực tiếp với u. **Yêu cầu:** Cho đồ thị và mảng b $(0 \leqslant b_i < p)$, hãy đếm số lượng mảng a thỏa mãn. Dữ liệu đảm bảo tồn tại ít nhất 1 nghiệm thỏa mãn.

INPUT

- Dòng đầu chứa 3 số nguyên n, m, p $(1 \leqslant n \leqslant 100, 1 \leqslant m \leqslant 200, p \leqslant 10^9)$ lần lượt là số đỉnh, số cạnh và số nguyên tố p.
- Dòng tiếp theo là n số nguyên của mảng b, mỗi số cách nhau 1 dấu cách.
- m dòng sau, mỗi dòng là 2 số u, v cho biết có cạnh nối giữa u và v.

OUTPUT

- Một số nguyên duy nhất là số lượng mảng a thỏa mãn lấy dư cho $10^9 + 7$.

SAMPLE TEST

Input:

```
5 5 13
5 20 0 23 5
1 4
2 1
2 4
4 5
5 2
```

Output:

```
2197
```

Phân tích

- Ta có thể đưa bài toán trở thành đếm số nghiệm của hệ phương trình sau:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \quad \dots \quad + a_{1m}x_m &\equiv b_1 \pmod p \\ a_{21}x_1 + a_{22}x_2 + \quad \dots \quad + a_{2m}x_m &\equiv b_2 \pmod p \\ &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + \quad \dots \quad + a_{nm}x_m &\equiv b_n \pmod p \end{aligned}$$

  với $a_{uv} = 1$ khi và chỉ khi giữa u và v có cạnh nối trực tiếp, ngược lại thì $a_{uv} = 0$.
- Giả sử hệ phương trình trên có k ẩn x là ẩn tự do, khi ta cố định cả k ẩn đó thì tất cả các ẩn khác cũng sẽ được cố định. $\rightarrow$ Với mỗi bộ k số để gán vào các ẩn tự do đó, ta sẽ tìm được đúng 1 nghiệm thỏa mãn.

  $\Longrightarrow$ Nếu ta tìm được k ẩn tự do thì đáp án sẽ là p^k.

Cài đặt

Bước 1: Tạo đồ thị. **Bước 2:** Tạo hệ phương trình. **Bước 3:** Khử các ẩn như ở trên, chỉ khác là ta có thêm biến `count` để đếm số lượng ẩn tự do. **Bước 4:** Vì dữ liệu đảm bảo tồn tại ít nhất 1 nghiệm thỏa mãn nên kết quả sẽ là p^{count}.

[41] https://hnoj.edu.vn/problem/findgraph

Ví dụ 2 – Codeforces - 1155E [42]

Đề bài

Có một đa thức bậc k $(k \leqslant 10)$ có hệ số nguyên:

$$f(x) = a_0 + a_1 \times x + a_2 \times x^2 + \\ + \cdots + a_k \times x^k \quad (0 \leqslant a_i < 10^6 + 3)$$

Ta được đưa ra tối đa 50 câu hỏi có dạng "? x_0" $(x_0 \in \mathbb{Z},\ 0 \leqslant x_0 < 10^6 + 3)$, với mỗi câu hỏi chương trình sẽ trả về giá trị của $f(x_0)$ mod $10^6 + 3$. Hãy tìm giá trị x_0 sao cho $f(x_0) \equiv 0 \pmod{10^6 + 3}$. Nếu tìm được x_0 thỏa mãn thì in ra "! x_0", còn không thì in ra "! -1".

Phân tích

- Nếu ta hỏi "? 0" thì ta sẽ được giá trị của a_0.
- Nếu ta hỏi k câu hỏi thì ta sẽ được hệ phương trình như sau (theo modulo $10^6 + 3$):

$$\begin{aligned} a_0 + a_1 x_1 + a_2 x_1^2 + \quad \dots \quad + a_k x_1^k &\equiv b_1 \\ a_0 + a_1 x_2 + a_2 x_2^2 + \quad \dots \quad + a_k x_2^k &\equiv b_2 \\ \vdots \\ a_0 + a_1 x_k + a_2 x_k^2 + \quad \dots \quad + a_k x_k^k &\equiv b_k \end{aligned}$$

Mà tất cả các ẩn $x_1, x_2, \dots, x_k$ ta đã biết nên bài toán trở thành giải hệ phương trình để tìm các hệ số a_i.

$\Longrightarrow$ Ta sẽ thử tìm đa thức $f(x)$ rồi thử với tất cả các giá trị x từ 0 đến $10^6 + 2$.

Cài đặt

Bước 1: Thử $x_0 = 0$ để tìm a_0. **Bước 2:** Hỏi khoảng 15 câu hỏi với các giá trị x_0 random từ 1 đến $10^6 + 2$. **Bước 3:** Với mỗi k từ 0 đến 10, ta sẽ lập hệ phương trình và tìm các hệ số a_i. Sau đó thử lại với cả 15 giá trị x_0 ở trên, nếu thỏa mãn thì đấy chính là đa thức cần tìm. Còn nếu tất cả k từ 0 đến 10 đến không thỏa mãn thì in ra "! -1". **Bước 4:** Thử thay tất cả các x từ 0 đến $10^6 + 2$ vào đa thức, nếu $f(x) \equiv 0 \pmod{10^6 + 3}$ thì kết quả chính là x.

[42] https://codeforces.com/contest/1155/problem/E

Ví dụ 3 – FSMAX - HNOJ [43]

Đề bài

Một tập các số được gọi là đẹp nếu không tồn tại một tập con nào của tập đó có tích là số chính phương. Cho dãy A có n số tự nhiên, hãy tìm tập con đẹp có nhiều phần từ nhất của A.

INPUT

- Dòng đầu chứa số nguyên dương n $(n \leqslant 1000)$.
- Dòng sau chứa n số tự nhiên của dãy A $(1 \leqslant A_i \leqslant 1000)$.

OUTPUT

- Một số nguyên duy nhất là số lượng phần từ lớn nhất của tập con.

SAMPLE TEST

Input:

```
5
8 2 1 6 7
```

Output:

```
3
```

Phân tích

- Số chính phương là số khi phân tích số đó thành tích các số nguyên tố thì tất cả các số mũ đều là chẵn. $\Longrightarrow$ Ta chỉ cần quan tâm đến tính chẵn lẽ của các số mũ nên có thể sử dụng bitset để lưu các giá trị A_i. Nếu ta đặt mỗi bit là một số nguyên tố thì chỉ cần 168 bit là được.
- Một tập các số có tích là số chính phương tức là XOR của tất cả các bitset của các số đó là 0.
- Hint: Nếu ta coi mỗi bitset là các hệ số của vế trái 1 phương trình thì kết quả chính là số lượng ẩn không phải ẩn tự do.

Cài đặt

- Cài đặt mẫu [44] .
- Bạn đọc hãy thử code trước khi xem code mẫu.

[43] https://hnoj.edu.vn/problem/fsmax

[44] https://ideone.com/SfwHl6

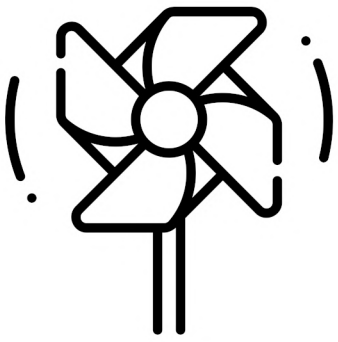
Một số bài tập ứng dụng

- SPOJ - XMAX [45]
- Codechef - KNGHTMOV [46]
- Codeforces - 1411G [47]
- Codeforces - 1060H [48]
- Codeforces - 1100F [49]
- Codeforces - 1101G [50]
- Codeforces - 1336E1 [51]
- Codeforces - 504D [52]

[45] https://www.spoj.com/problems/XMAX/
[46] https://www.codechef.com/SEP12/problems/KNGHTMOV
[47] https://codeforces.com/problemset/problem/1411/G
[48] https://codeforces.com/contest/1060/problem/H
[49] https://codeforces.com/contest/1100/problem/F
[50] https://codeforces.com/contest/1101/problem/G
[51] https://codeforces.com/contest/1336/problem/E1
[52] https://codeforces.com/contest/504/problem/D

Kỹ thuật tinh tế về phép Xor

Nguyễn Minh Thiện
Sinh viên năm 2, Trường Công Nghệ Thông Tin và Truyền Thông, Đại học Cần Thơ



Giới thiệu

Đây là một kỹ thuật được dùng để giải quyết một số bài toán có liên quan đến tổng xor của đoạn con (liên tiếp hoặc không liên tiếp) của mảng các số nguyên cho trước hoặc thêm phần tử vào mảng song song với việc truy vấn.

Kỹ thuật này có thể được chia làm hai phần chính:

- Biểu diễn các số nguyên ở cơ số 2 và xem mỗi phần tử là một vector trong không gian vector $\mathbb{Z}_2^d$, với d là số bit tối đa cần dùng để biểu diễn. Lúc này phép xor giữa các phần tử tương đương với phép cộng giữa các vector tương ứng trong không gian vector $\mathbb{Z}_2^d$.
- Tìm mối liên hệ giữa yêu cầu của các truy vấn và cơ sở của không gian vector tìm được ở trên.

Một số khái niệm

Không gian Vector

Một tập $V \neq \varnothing$ được gọi là một không gian vector nếu V được trang bị hai phép toán: phép cộng và phép nhân với một vô hướng. Các phần tử trong V được gọi là các vector.

- V phải có tính chất đóng, tức là $\forall x, y \in V \Longrightarrow x+y \in V$ và $\forall x \in V \Longrightarrow kx \in V$ với k là một đại lượng vô hướng.
- Phép cộng phải có tính chất kết hợp và giao hoán.
- Phép nhân với một vô hướng phải có tính chất kết hợp và phân phối.
- Trong V phải có phần tử O gọi là vector không (chú ý không lẫn với số 0).

Một số không gian vector đặc biệt:

- Không gian vector các tọa độ $\mathbb{R}^n = \{(x_1, x_2, \ldots, x_n)\}$.
- Không gian vector $M_{n\times m}$ các ma trận kích cỡ $n \times m$.
- Không gian vector $P_n(x)$ các đa thức bậc không quá n.
- Không gian vector F các hàm số.
- ...

Độc lập tuyến tính và phụ thuộc tuyến tính

Một tập các vector $\{v_1, v_2, \ldots, v_n\}$ được gọi là **độc lập tuyến tính** nếu phương trình:

$$x_1v_1 + x_2v_2 + \ldots + x_nv_n = O$$

chỉ có duy nhất một nghiệm tầm thường $x_1 = x_2 = \ldots = x_n = 0$.

Ngược lại nếu tồn tại $x_1, x_2, \ldots, x_n$ không đồng thời bằng không, sao cho:

$$x_1v_1 + x_2v_2 + \ldots + x_nv_n = O$$

thì khi đó được gọi là **phụ thuộc tuyến tính**.

Không gian vector $\mathbb{Z}_2^d$

$\mathbb{Z}_2$: $\mathbb{Z}_m$ là tập các số dư khi chia lấy dư cho m và các phép toán trên $\mathbb{Z}_m$ cũng chia lấy dư cho m. Suy ra $\mathbb{Z}_2$ là tập các số dư khi chia lấy dư cho 2 do đó $\mathbb{Z}_2 = \{0, 1\}$.

$\mathbb{Z}_2^d$: là một không gian vector d chiều bao gồm tất cả các vector có d tọa độ, mỗi tọa độ là một phần tử của $\mathbb{Z}_2$.

- Nếu hai vector $x, y \in \mathbb{Z}_2^d$ thì $x + y$ được định nghĩa là $x \oplus y$ ($\oplus$ là phép xor bit). Chú ý $x + y \in \mathbb{Z}_2^d$ phải thõa mãn. Cụ thể hơn:

		$\oplus$	**Tổng**	**Tổng mod 2**
0	0	0	0	0
0	1	1	1	1
1	0	1	1	1
1	1	0	2	0

- Với $c \in \mathbb{Z}_2$, kí hiệu $cx = \underbrace{x + x + \ldots + x}_{c \text{ lần } x}$. Ta có $cx = 0$ nếu c chẵn và $cx = x$ nếu c lẻ.

Kể từ đây chúng ta chỉ quan tâm đến không gian vector $\mathbb{Z}_2^d$, mọi phép tính đều chia lấy dư cho 2 và xem phép $\oplus$ (xor bit) tương đương với phép $+$ trong $\mathbb{Z}_2^d$.

Bao tuyến tính (Linear span, gọi tắt là span)

Một không gian vector V được span bởi một tập các vector $S = \{v_1, v_2, \ldots, v_n\}$ chứa tất cả các vector x được biểu diễn thông qua một tổ hợp tuyến tính của các vector trong S.

$$\text{span}(S) = \left\{ \sum_{i=1}^{n} c_i v_i \;\middle|\; v_i \in S, c_i \in \{0, 1\} \right\}$$

lúc này ta nói V được span bởi S, V được sinh bởi S hay S sinh ra V.

Chú ý: do chỉ xét trên $\mathbb{Z}_2^d$ nên c_i cũng phải thuộc $\mathbb{Z}_2^d$. Do đó trong tổ hợp trên mỗi v_i chỉ có hai trạng thái là **xuất hiện** hoặc **không xuất hiện**.

Ví dụ: tìm span$(\{2,5\})$, span$(\{2,5,7\})$ và span$(\{\})$?

- Ta có $2 \oplus 5 = 7$, suy ra:

$$\text{span}(\{2,5\}) = \text{span}(\{2,5,7\}) = \{0,2,5,7\}$$
$$\text{span}(\{\}) = \{0\}$$

Một số tính chất quan trọng:

- Nếu $v_{n+1} \in \text{span}(\{v_1, v_2, \ldots, v_n\})$ thì: $\text{span}(\{v_1, v_2, \ldots, v_n, v_{n+1}\}) = \text{span}(\{v_1, v_2, \ldots, v_n\})$
- $\text{span}(\{v_1 + v_3, v_2, \ldots, v_n\}) = \text{span}(\{v_1, v_2, \ldots, v_n\})$

Cơ sở (Basis)

Một tập các vector $B = \{v_1, v_2, \ldots, v_n\}$ được gọi là cơ sở của một không gian vector V nếu $\text{span}(B) = V$ và B là **độc lập tuyến tính**. Khi đó n được gọi là **số chiều** của V và kí hiệu là $\dim(V)$.

Ví dụ: Xét không gian vector $V = \{0,2,5,7\}$, ta có $\{2,5\}$ là một cơ sở của V nhưng $\{2,5,7\}$ thì không vì tập này là không độc lập tuyến tính do $2 \oplus 5 \oplus 7 = 0$. Và dĩ nhiên $\{2,7\}$ và $\{5,7\}$ cũng là các cơ sở của V.

Chú ý: số phần tử phân biệt trong không gian vector V được span bởi một cơ sở $B = \{v_1, v_2, \ldots, v_n\}$ là $|V| = 2^n$.

Ma trận

Không gian hàng và không gian cột

Xét một ma trận M kích cỡ $n \times m$ (n hàng và m cột):

$$M = \begin{bmatrix} v_{1,1} & v_{1,2} & \cdots & v_{1,m} \\ v_{2,1} & v_{2,2} & \cdots & v_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ v_{n,1} & v_{n,2} & \cdots & v_{n,m} \end{bmatrix}$$

- Chia ma trận M thành các vector hàng, ta được:

$$row_i = [v_{i,1}, v_{i,2}, \ldots, v_{i,m}]$$

- Chia ma trận M thành các vector cột, ta được:

$$col_i = [v_{1,i}, v_{2,i}, \ldots, v_{n,i}]$$

- Khi đó không gian hàng và không gian cột của ma trận M được định nghĩa như sau:

$$RS(M) = \text{span}(\{row_1, row_2, \ldots, row_n\})$$
$$CS(M) = \text{span}(\{col_1, col_2, \ldots, col_m\})$$

Hạng của ma trận

Người ta chứng minh được rằng $\dim(RS(M)) = \dim(CS(M))$ và được kí hiệu là $\dim(M)$ hay còn được biết đến là hạng (rank) của ma trận M, kí hiệu $\text{rank}(M)$. (Tham khảo phần chứng minh tại đây [53])

Không gian hạch (Null space)

Để hiểu rõ được phần này, bạn cần phải quen thuộc với phép nhân ma trận [54] .

Không gian hạch của một ma trận M, kí hiệu là $\text{null}(M)$ là tập các vector $x \in \mathbb{R}^m$ (được viết ở dạng cột) sao cho:

$$M \cdot x = O$$

Giả sử:

$$M = \begin{bmatrix} v_{1,1} & v_{1,2} & \cdots & v_{1,m} \\ v_{2,1} & v_{2,2} & \cdots & v_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ v_{n,1} & v_{n,2} & \cdots & v_{n,m} \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ x_m \end{bmatrix}$$

Khi đó $(x_1, x_2, \ldots, x_m)$ là nghiệm của hệ phương trình:

$$\begin{cases} v_{1,1}x_1 + v_{1,2}x_2 + \ldots + v_{1,m}x_m = 0 \\ v_{2,1}x_1 + v_{2,2}x_2 + \ldots + v_{2,m}x_m = 0 \\ \ldots \\ v_{n,1}x_1 + v_{n,2}x_2 + \ldots + v_{n,m}x_m = 0 \end{cases}$$

Đặc biệt: nếu ma trận M kích cỡ $n \times m$ gồm m vector $v \in \mathbb{Z}_2^n$ và vector $x \in \mathbb{Z}_2^m$ (các vector v

[53] https://math.stackexchange.com/questions/332908/looking-for-an-intuitive-explanation-why-the-row-rank-is-equal-to-the-column-ran

[54] https://vi.wikipedia.org/wiki/Ph%C3%A9p_nh%C3%A2n_ma_tr%E1%BA%ADn

và x được viết ở dạng cột):

$$\begin{aligned} M &= \begin{bmatrix} | & | & \dots & | \\ v_1 & v_2 & \dots & v_m \\ | & | & \dots & | \end{bmatrix} \\ &\equiv \begin{bmatrix} v_{1,1} & v_{2,1} & \cdots & v_{m,1} \\ v_{1,2} & v_{2,2} & \cdots & v_{m,2} \\ \vdots & \vdots & \ddots & \vdots \\ v_{1,n} & v_{2,n} & \cdots & v_{m,n} \end{bmatrix} \\ x &= \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_m \end{bmatrix} \end{aligned}$$

Khi đó:

$$M \cdot x = O$$
$$\iff \begin{cases} v_{1,1}x_1 + v_{2,1}x_2 + \ldots + v_{m,1}x_m \equiv 0 \\ v_{1,2}x_1 + v_{2,2}x_2 + \ldots + v_{m,2}x_m \equiv 0 \\ \dots \\ v_{1,n}x_1 + v_{2,n}x_2 + \ldots + v_{m,n}x_m \equiv 0 \end{cases}$$

Lưu ý các phương trình trong hệ phương trình trên là đồng dư theo **modulo** 2.

Để ý rằng không gian hạch cũng là một không gian vector, vì:

$$\begin{cases} M \cdot a = O \\ M \cdot b = O \end{cases} \implies M \cdot (a+b) = O$$
$$M \cdot a = O \implies M \cdot (ca) = O$$

Số vô hiệu (Nullity)

Số vô hiệu của một ma trận M chính là số chiều của không gian hạch của M:

$$\text{nullity}(M) = \dim(\text{null}(M))$$

Định lý về hạng và số vô hiệu

Định lý này phát biểu rằng với một ma trận M kích cỡ $n \times m$ thì:

$$\text{rank}(M) + \text{nullity}(M) = \text{số cột} = m$$

Phần chứng minh khá phức tạp và đòi hỏi giới thiệu thêm nhiều khái niệm nên các bạn có thể tham khảo tại đây [55] .

Thuật toán tìm cơ sở của một không gian vector

Tiếp theo chúng ta sẽ đi vào phần chính của bài viết này đó chính là làm thế nào để có thể tìm cơ sở của một không gian vector, trong đó mỗi vector là một phần tử của $\mathbb{Z}_2^d$ một cách hiệu quả. Chúng ta sẽ phân tích thuật toán thông qua bài toán dưới đây.

XOR Closure [56]

Đề bài

Cho một mảng a có n phần tử phân biệt $a_1, a_2, \ldots, a_n$. Yêu cầu tìm số phần tử ít nhất cần thêm vào mảng a sao cho điều sau luôn đúng: với mọi x, y thuộc a thì $x \oplus y$ cũng thuộc a.

Giới hạn

- $1 \le n \le 10^5$.
- $0 \le a_i \le 10^{18}$.

Lời giải

Để ý rằng chúng ta cần thỏa mãn điều kiện $\forall x, y \in a$ thì $x \oplus y \in a$. Do đó cần phải xây dựng được **cơ sở** B của không gian vector V được span bởi mảng a. Khi đó đáp án của bài toán chính là $2^{\dim(\text{span}(a))} - n = 2^{|B|} - n$.

Chúng ta sẽ xét lần lượt từng phần tử như sau:

- Giả sử chúng ta đã xét đến $a_1, a_2, \ldots, a_{i-1}$ và có cơ sở B. Chúng ta cần cập nhật B sao cho a_i cũng có thể được biểu diễn thông qua các vector trong B.
- Kiểm tra xem a_i có thể được biểu diễn thông qua các vector trong B hay không.
  - Nếu có thì không cần làm gì cả.
  - Ngược lại chỉ cần thêm a_i vào B.

Phần khó nhất đó chính là làm thế nào để có thể kiểm tra xem a_i có thể được biểu diễn thông qua các vector trong B hay không.

- Nếu xem xét tất cả 2^i tổ hợp tuyến tính của các vector trong B thì rõ ràng sẽ không đủ thời gian.
- Với một số nguyên dương x, định nghĩa $\text{msb}(x)$ = vị trí của most significant bit trong x. Ví dụ:

$$\text{msb}(5) = \text{msb}(7) = 2\text{msb}(3) = 1, \text{msb}(1) = 0$$

- Giả sử chúng ta có cơ sở $B = \{b_1, b_2, \ldots, b_k\}$ của $a_1, a_2, \ldots, a_{i-1}$ sao cho $\text{msb}(b_1) < \text{msb}(b_2) < \ldots < \text{msb}(b_k)$.

[55] https://en.wikipedia.org/wiki/Rank%E2%80%93nullity_theorem#First_proof
[56] https://csacademy.com/contest/archive/task/xor-closure/

- Khi đó chỉ cần duyệt j từ $1..k$. Nếu $a_i + b_j < a_i$ ($\text{msb}(a_i) = \text{msb}(b_j)$) thì thay $a_i = a_i + b_j$. Nếu cuối cùng $a_i = 0$ thì a_i có thể được biểu diễn thông qua các vector trong B. Ngược lại thêm giá trị hiện tại của a_i vào vị trí thích hợp trong B sao cho vẫn giữ được thứ tự giảm dần của msb.
- **Ví dụ:** kiểm tra xem $x = 20$ có thuộc $\text{span}(\{26, 15, 3, 1\})$ hay không?

$$\begin{aligned} 26 &= 11010_2 \\ 15 &= 01111_2 \\ 3 &= 00011_2 \\ 1 &= 00001_2 \\ 20 &= 10100_2 \end{aligned}$$

  - $x \oplus 26 = \mathbf{14} < x \Rightarrow$ cập nhật $x = 14$.
  - $x \oplus 15 = \mathbf{1} < x \Rightarrow$ cập nhật $x = 1$.
  - $x \oplus 3 = \mathbf{2} > x$.
  - $x \oplus 1 = \mathbf{0} < x \Rightarrow$ cập nhật $x = 0$.

  Vậy $20 = 26 \oplus 15 \oplus 1$ và $20 \in \text{span}(\{26, 15, 3, 1\})$.
- **Tổng quát:** thuật toán trên cho phép tìm giá trị nhỏ nhất của $(a_i + v)$ với $v \in \text{span}(\{b_1, b_2, \ldots, b_k\})$.
- **Ví dụ:** xét $x = 9$ và $\text{span}(\{26, 15, 3, 1\})$. Khi đó giá trị nhỏ nhất có thể đạt được là:

$$9 \oplus 15 \oplus 3 \oplus 1 = \mathbf{4} = 100_2.$$

Code mẫu:

```
vector<long long> basis;
int sz;
void insertVector(long long mask) {
  // duyệt các phần tử theo thứ tự giá trị msb
  // giảm dần.
  for (int i = 0; i < sz; ++i) {
    mask = min(mask, mask ^ basis[i]);
  }
  if (mask != 0) {
    basis.push_back(mask);
    sz++;
    int i = sz - 1;
    // giữ các giá trị msb sao cho vẫn theo thứ tự
    // giảm dần.
    while (i > 0 && basis[i - 1] < basis[i]) {
      swap(basis[i - 1], basis[i]);
      i--;
    }
  }
}
```

Phân tích độ phức tạp

Vì các phần tử có kiểu `long long` nên cần tối đa 64 bit để biểu diễn. Đặt $d = basis.size() = \dim(\text{span}(a)) \leq 64$ thì độ phức tạp mỗi lần thêm một vector là $\mathcal{O}(d)$. Độ phức tạp tổng cộng cho việc thêm n vector là $\mathcal{O}(nd)$.

Để ý rằng trong thuật toán trên v_i chỉ được thêm vào mảng `basis` khi không có phần tử b_j nào mà $\text{msb}(b_j) = \text{msb}(v_i)$. Do đó các giá trị $\text{msb}(b_j)$ trong B là phân biệt. Khi đó có thể cải tiến code như sau:

Code cải tiến

```
vector<long long> basis;
void insertVector(long long mask) {
  for (int i = 0; i < (int)basis.size(); ++i) {
    mask = min(mask, mask ^ basis[i]);
  }
  if (mask != 0) {
    basis.push_back(mask);
    // chỉ cần thêm vào cuối mảng, không cần giữ
    // giá trị msb theo thứ tự giảm dần.
  }
}
```

Code hoàn chỉnh

```
#include <bits/stdc++.h>
using namespace std;

vector<long long> basis;
void insertVector(long long mask) {
  for (int i = 0; i < (int)basis.size(); ++i) {
    mask = min(mask, mask ^ basis[i]);
  }
  if (mask != 0) { basis.push_back(mask); }
}

int main() {
  ios::sync_with_stdio(false);
  cin.tie(nullptr);
  int n;
  cin >> n;
  vector<long long> a(n);
  for (int i = 0; i < n; ++i) {
    cin >> a[i];
    insertVector(a[i]);
  }
  long long ret = (1LL << basis.size()) - n;
  cout << ret << '\n';

  return 0;
}
```

Một số bài tập ví dụ

Codechef - XORCMPNT [57]

Tóm tắt đề bài

Cho 2^K trạm điện (được đánh số từ 0 đến $2^K - 1$). Ban đầu không có đường nối giữa các trạm. Bạn được cho M số nguyên $x_1, x_2, \ldots, x_M$, giữa hai trạm u, v khác nhau sẽ có một đường nối trực tiếp nếu tồn tại chỉ số i thỏa mãn $u \oplus v = x_i$. Hai trạm u, v được gọi là cùng một thành phần liên thông nếu chúng có đường nối trực tiếp với nhau hoặc gián tiếp qua các trạm khác. Bạn cần xử lý T trường hợp. Mỗi trường hợp yêu cầu tìm số thành phần liên thông được tạo thành.

Giới hạn

- $1 \leq T \leq 10^5$.

[57] https://www.codechef.com/problems/XORCMPNT

- $1 \leq K \leq 30$.
- $1 \leq M \leq 10^5$.
- $0 \leq x_i < 2^K$.
- Tổng M trong tất cả các trường hợp không quá 10^5.

Lời giải

- Do $K \leq 30$ nên chúng ta không thể xem xét hết tất cả 2^K trạm.
- Giả sử chúng ta có ba trạm như sau: $u \oplus v = x_i, v \oplus w = x_j$. Khi đó hai trạm u và w sẽ có đường đi gián tiếp qua trạm v và thuộc cùng một thành phần liên thông và $u \oplus w = x_i \oplus x_j$.
- Do đó hai trạm u, v bất kì sẽ thuộc cùng một thành phần liên thông nếu $u \oplus v = x_i \oplus x_{i+1} \oplus \ldots \oplus x_j$, với $\{x_i, x_{i+1}, \ldots, x_j\}$ là một tập con bất kì của $\{x_1, x_2, \ldots, x_M\}$. Đặt $z = x_i \oplus x_{i+1} \oplus \ldots \oplus x_j$, suy ra $z \in \text{span}(\{x_1, x_2, \ldots, x_M\})$ và đặt $d = \dim(\text{span}(\{x_1, x_2, \ldots, x_M\}))$.
- Xét một trạm u bất kì, ta có các trạm cùng một thành phần liên thông với u là v $(v \neq u)$ sao cho $u \oplus v = z \Longrightarrow v = u \oplus z$. Có tất cả 2^d giá trị z khác nhau (bao gồm cả $z = 0$ và $v = u$) nên sẽ có tất cả $2^d - 1$ trạm v như vậy.
- Như vậy mỗi trạm u bất kì sẽ liên thông với $2^d - 1$ trạm khác. Cho nên mỗi thành phần liên thông đều có 2^d trạm và do đó sẽ có tất cả $2^K / 2^d$ thành phần liên thông.

Code mẫu

```
#include <bits/stdc++.h>
using namespace std;

vector<int> basis;
void insertVector(int mask) {
  for (int i = 0; i < (int)basis.size(); ++i) {
    mask = min(mask, mask ^ basis[i]);
  }
  if (mask != 0) { basis.push_back(mask); }
}

int main() {
  ios::sync_with_stdio(false);
  cin.tie(nullptr);
  int t;
  cin >> t;
  while (t--) {
    int k, m;
    cin >> k >> m;
    vector<int> a(m);
    for (int i = 0; i < m; ++i) { cin >> a[i]; }
    // xây dựng mảng cơ sở của không gian vector
    // được span bởi {x_1, x_2,..., x_m}.
    basis.clear();
    for (int i = 0; i < m; ++i) {
      insertVector(a[i]);
    }
    int d = (int)basis.size();
    cout << (1 << k) / (1 << d) << '\n';
  }
  return 0;
}
```

Phân tích độ phức tạp

Hàm `insertVector` có độ phức tạp là $\mathcal{O}(d)$, với d là số chiều của không gian vector được span bởi $\{x_1, x_2, \ldots, x_M\}$ (kích thước của mảng `basis`) trong trường hợp này $x_i < 2^K$ nên $d \leq K$. Độ phức tạp tổng cộng $\mathcal{O}(MK)$.

Maximum XOR over all subsets [58]

Đề bài

Cho tập S có n phần tử $a_1, a_2, \ldots, a_n$. Hãy cho biết tổng XOR lớn nhất trong tất cả các tập con của S là bao nhiêu?

Giới hạn

- $1 \leq n \leq 10^5$.
- $0 \leq a_i < 2^{20}$.

Lời giải

- Do n khá lớn nên ta không thể xét hết tất cả 2^n tập con của S.
- Nhận xét rằng do $a_i < 2^{20}$ nên ta có thể tìm cơ sở B của không gian vector V được span bởi S. Khi đó chỉ cần duyệt hết tất cả $2^{|B|}$ tập con của B ($|B| \leq 20$). Độ phức tạp $\mathcal{O}(2^{|B|})$.

Chứng minh

- Giả sử $B = \{b_1, b_2, \ldots, b_k\}$.
- Do mỗi phần tử trong tập S đều được biểu diễn thông qua các phần tử trong B ($a_i = \sum_{i=1}^{k} c_i b_i, c_i \in \{0, 1\}$) nên xét một tập con bất kì $\{a_1, a_2, \ldots, a_m\}$, khi đó $a_1 + a_2 + \ldots + a_m$ cũng sẽ biểu diễn được thông qua các phần tử trong B.

Code mẫu

```
int k = (int)basis.size();
int answer = 0;
// duyệt qua tất cả 2^k tập con.
for (int mask = 0; mask < (1 << k); ++mask) {
  int cur =
    0; // lưu tổng xor của tập con hiện tại.
  for (int i = 0; i < k; ++i) {
    if (mask & (1 << i)) { cur ^= basis[i]; }
  }
  answer = max(answer, cur);
}
cout << answer << '\n';
```

Tối ưu hơn

- Chúng ta có thể dùng ý tưởng tham lam bằng cách ưu tiên set cho bit có `index` lớn hơn trong `res` bằng 1. Vì các bit phía sau dù đều bằng 1 thì vẫn sẽ không tối ưu nếu bit i bằng 0: $2^0 + 2^1 + \ldots + 2^{i-1} = 2^i - 1 < 2^i$. Độ phức tạp $\mathcal{O}(|B|)$.

[58] https://codeforces.com/blog/entry/60003

```cpp
int k = (int)basis.size();
// không cần duyệt theo thứ tự giảm dần msb do các
// giá trị msb trong mảng basis là phân biệt.
int answer = 0;
for (int i = 0; i < k; ++i) {
  answer = max(answer, answer ^ basis[i]);
}
cout << answer << '\n';
```

Codeforces 895C - Square subsets [59]

Tóm tắt đề bài

Cho một mảng a gồm n số nguyên dương. Tìm số cách khác nhau để chọn ra từ mảng a một tập con khác rỗng sao cho tích của các phần tử được chọn là một số chính phương. Hai cách chọn được coi là khác nhau nếu tồn tại một vị trí được chọn bởi tập này mà không được chọn bởi tập kia. Do đáp án có thể rất lớn nên bạn cần in ra đáp án sau khi chia lấy dư cho $10^9 + 7$.

Giới hạn

- $1 \le n \le 10^5$.
- $1 \le a_i \le 70$.

Lời giải

Bài này có thể giải bằng quy hoạch động bitmask. Nhưng mình sẽ giới thiệu một cách tiếp cận khác đơn giản hơn và độ phức tạp thấp hơn, áp dụng **định lí về hạng và số vô hiệu** của ma trận.

- Ta biết rằng mọi số nguyên dương x bất kì đều được biểu diễn thành tích các thừa số nguyên tố $p_1^{k_1} \cdot p_2^{k_2} \cdots p_m^{k_m}$ (trong đó p_i là các số nguyên tố, $k_i \ge 0$). x là số chính phương khi và chỉ khi các số $k_1, k_2, \ldots, k_m$ đều là số chẵn.
- Với mỗi số, chúng ta chỉ cần quan tâm các vị trí i mà k_i là số lẻ. Chỉ có 19 số nguyên tố trong đoạn $[1, 70]$ nên có thể coi mỗi số là một vector trong $\mathbb{Z}_2^{19}$ và phép nhân hai số tương đương với phép cộng hai vector tương ứng. Khi đó số chính phương chính là một vector không (O).
- Đặt v_i là vector tương ứng của a_i trong $\mathbb{Z}_2^{19}$. Khi đó ta biểu diễn lại mảng a thành một ma trận kích cỡ $19 \times n$ như sau (các vector v_i được viết ở dạng cột):

$$M = \begin{bmatrix} | & | & \dots & | \\ v_1 & v_2 & \dots & v_n \\ | & | & \dots & | \end{bmatrix}$$

- Chúng ta cần tìm tất cả các vector $x = [x_1, x_2, \ldots, x_n]$ $(x \in \mathbb{Z}_2^n)$, trong đó $x_i = 0$ tương ứng với a_i không được chọn và $x_i = 1$ tương ứng với a_i được chọn, sao cho:

$$M \cdot x = O$$

- Số vector x thỏa mãn chính bằng số vector trong không gian hạch của M và bằng $2^{\text{nullity}(M)}$ (bao gồm cả cách chọn tập rỗng từ a). Khi đó đáp án chính là $2^{\text{nullity}(M)} - 1$.
- Theo định lí về hạng và số vô hiệu:

$$\text{rank}(M) + \text{nullity}(M) = \text{số cột} = n$$

Dễ dàng suy ra:

$$\begin{aligned} \text{nullity}(M) &= n - \text{rank}(M) \\ &= n - \dim(\text{CS}(M)) \\ &= n - \dim(\text{span}(a)) \end{aligned}$$

Code mẫu

```cpp
#include <bits/stdc++.h>
using namespace std;

const int MOD = (int)1e9 + 7;

vector<int> primes, basis;
const int MAX_VAL = 70;

void precompute() {
  // Tìm các số nguyên tố <= 70.
  for (int i = 2; i <= MAX_VAL; ++i) {
    bool is_prime = true;
    for (int j = 2; j * j <= i; ++j) {
      if (i % j == 0) {
        is_prime = false;
        break;
      }
    }
    if (is_prime) { primes.push_back(i); }
  }
}

void insertVector(int mask) {
  for (int i = 0; i < (int)basis.size(); ++i) {
    mask = min(mask, mask ^ basis[i]);
  }
  if (mask != 0) { basis.push_back(mask); }
}

int main() {
  ios::sync_with_stdio(false);
  cin.tie(nullptr);

  precompute();
  int n;
  cin >> n;
  vector<int> a(n), cnt(MAX_VAL + 1);
  for (int i = 0; i < n; ++i) {
    cin >> a[i];
    cnt[a[i]]++;
  }
  // Tìm vector v tương ứng.
  for (int i = 0; i <= MAX_VAL; ++i) {
    if (!cnt[i]) continue;
    int v = 0, cur = i;
    for (int j = 0; j < (int)primes.size(); ++j) {
      int k = 0;
      while (cur % primes[j] == 0) {
        cur /= primes[j];
        k++;
```

[59] https://codeforces.com/contest/895/problem/C

```
      }
      // Nếu k lẻ thì đặt bit thứ j bằng 1.
      if (k & 1) { v += (1 << j); }
    }
    insertVector(v);
  }
  int d = (int)basis.size();
  int nullity = n - d;
  int answer = 1;
  for (int i = 0; i < nullity; ++i) {
    answer = answer * 2 % MOD;
  }
  cout << (answer == 0 ? MOD - 1 : answer - 1)
       << '\n';
  return 0;
}
```

Độ phức tạp

- Đặt $m = max\{a_i\}, d = basis.size()$.
- Độ phức tạp là $\mathcal{O}(m\sqrt{m} + md \log m + n) = \mathcal{O}(n)$.

Tổng quát

Cho một mảng a gồm n phần tử và một số nguyên s. Tìm số tập con của a sao cho tổng xor các phần tử đúng bằng s. Giới hạn: $1 \le n \le 10^5, 0 \le a_i < 2^{31}, 0 \le s < 2^{31}$.

Phân tích

- Coi mỗi phần tử a_i là một vector thuộc không gian vector $\mathbb{Z}_2^d$. Biểu diễn lại a bằng một ma trận M kích cỡ $d \times n$ như sau:

$$M = \begin{bmatrix} | & | & \dots & | \\ a_1 & a_2 & \dots & a_n \\ | & | & \dots & | \end{bmatrix} \equiv \begin{bmatrix} a_{1,1} & a_{2,1} & \cdots & a_{n,1} \\ a_{1,2} & a_{2,2} & \cdots & a_{n,2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1,d} & a_{2,d} & \cdots & a_{n,d} \end{bmatrix}$$

- Đầu tiên ta cần kiểm tra xem s có thể được biểu diễn thông qua các vector a_i hay không. Nếu không thì đáp án là 0, ngược lại giả sử $s = a_i + a_{i+1} + \ldots + a_j$ (với $\{a_i, a_{i+1}, \ldots, a_j\}$ là một tập con bất kì của a).
- Khi đó ta cần tìm số vector $x = [x_1, x_2, \ldots, x_n]$ ($x \in \mathbb{Z}_2^n$) sao cho:

$$\begin{aligned} & M \cdot x = s \\ \Leftrightarrow & \begin{cases} a_{1,1}x_1 + a_{2,1}x_2 + \ldots + a_{n,1}x_n \\ \quad \equiv a_{i,1} + a_{i+1,1} + \ldots + a_{j,1} \pmod 2 \\ a_{1,2}x_1 + a_{2,2}x_2 + \ldots + a_{n,2}x_n \\ \quad \equiv a_{i,2} + a_{i+1,2} + \ldots + a_{j,2} \pmod 2 \\ \vdots \\ a_{1,d}x_1 + a_{2,d}x_2 + \ldots + a_{n,d}x_n \\ \quad \equiv a_{i,d} + a_{i+1,d} + \ldots + a_{j,d} \pmod 2 \end{cases} \end{aligned}$$

Bằng phép biến đổi tương đương ta có được đáp án tương tự.

(Zero XOR Subset)-less [60]

Đề bài

Cho một mảng các số nguyên a gồm n phần tử. Nhiệm vụ của bạn là chia mảng đã cho thành nhiều nhất các đoạn, sao cho:

- Mỗi phần tử chỉ thuộc một đoạn.
- Mỗi đoạn chứa ít nhất một phần tử.
- Không tồn tại một tập khác rỗng các đoạn sao cho tổng xor các phần tử bằng 0.

In ra số đoạn nhiều nhất có thể chia thành. Hoặc -1 nếu không tồn tại cách chia.

Giới hạn

- $1 \le n \le 2 \cdot 10^5$.
- $0 \le a_i \le 10^9$.

Phân tích

- Ta sẽ xem tổng xor của các phần tử trong mỗi đoạn là một vector trong $\mathbb{Z}_2^{30}$.
- Dễ thấy rằng trong cách chia của đáp án thì tập các vector phải là độc lập tuyến tính.
- Giả sử ta có một cách chia thỏa mãn như sau (với số đoạn là nhiều nhất có thể): $[l_1 = 1, r_1], [l_2 = r_1 + 1, r_2], \ldots, [l_m = r_{m-1} + 1, r_m = n]$. Gọi v_i là tổng xor của đoạn thứ i. Khi đó ta có tập $S = \{v_1, v_2, \ldots, v_m\}$ là độc lập tuyến tính và $\dim(\text{span}(\mathbf{S})) = \mathbf{m}$.
- Theo tính chất của tập độc lập tuyến tính thì tập $Q = \{v_1, v_1 + v_2, \ldots, v_1 + v_2 + \ldots + v_m\} = \{p_{r_1}, p_{r_2}, \ldots, p_{r_m}\}$ (với p_i là tổng xor của i phần tử đầu tiên) cũng độc lập tuyến tính và $\dim(\text{span}(Q)) = \dim(\text{span}(S))$.
- Do cách chia trên đã là tối ưu nên ta có thể thêm bất kì p_i nào vào Q mà vẫn giữ nguyên số vector độc lập tuyến tính ($\dim(\text{span}(Q))$ không đổi).
- Đáp án cuối cùng là $\dim(\text{span}(\{p_1, p_2, \ldots, p_n\}))$. Ngoại trừ trường hợp $p_n = 0$ thì không có cách chia nào nên in ra -1.

Code mẫu

```
#include <bits/stdc++.h>
using namespace std;

vector<int> basis;
void insertVector(int mask) {
  for (int i = 0; i < (int)basis.size(); ++i) {
    mask = min(mask, mask ^ basis[i]);
  }
  if (mask != 0) { basis.push_back(mask); }
}
```

[60] https://codeforces.com/contest/1101/problem/G

```cpp
int main() {
  ios::sync_with_stdio(false);
  cin.tie(nullptr);
  int n;
  cin >> n;
  vector<int> a(n);
  int p =
    0; // lưu tổng xor của i phần tử đầu tiên.
  for (int i = 0; i < n; ++i) {
    cin >> a[i];
    p ^= a[i];
    insertVector(p);
  }
  // nếu p_n = 0 thì không có đáp án.
  if (p == 0) {
    cout << -1 << '\n';
  } else {
    cout << basis.size() << '\n';
  }
  return 0;
}
```

Độ phức tạp

Độ phức tạp là $\mathcal{O}(nd)$ (với $a_i \leq 10^9$ thì $d = basis.size() \leq 30$).

Bài tập áp dụng

- Codeforces[61] - Godzilla and Pretty XOR [62] (bạn cần tham gia nhóm tại đây [63])
- Codeforces - Round 473 - Div.2 - F [64]
- Atcoder - Xor Battle [65]
- Atcoder - Xor Sum 3 [66]
- Atcoder - Spices [67]
- Codeforces - Global round 11 - E. Xum [68]
- Hackerearth - Chef & Chutneys [69]
- Atcoder - Xor Query [70]
- Codeforces - Round 635 - Div.1 - E1 [71]

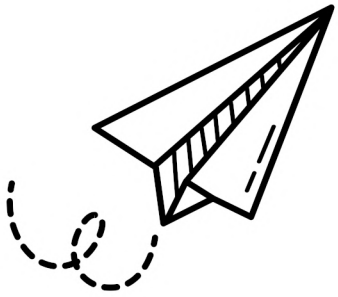
Các nguồn tham khảo

- Codeforces blog - A Beautiful Technique for Some XOR Related Problems [72]
- Codeforces blog - 2 Special cases of Gaussian [73]
- Benjamin Qi - Vector in $\mathbb{Z}_2^d$ [74]
- USACO - Xor basis [75]

[61] Codeforces: một trang web tổ chức các cuộc thi lập trình thi đấu với các dạng đề, bài tập đa dạng.
[62] https://codeforces.com/group/qcIqFPYhVr/contest/203881/problem/S
[63] https://codeforces.com/group/qcIqFPYhVr
[64] https://codeforces.com/contest/959/problem/F
[65] https://atcoder.jp/contests/agc045/tasks/agc045_a
[66] https://atcoder.jp/contests/abc141/tasks/abc141_f
[67] https://atcoder.jp/contests/abc236/tasks/abc236_f
[68] https://codeforces.com/problemset/problem/1427/E
[69] https://www.hackerearth.com/problem/algorithm/chef-f59c8115/
[70] https://atcoder.jp/contests/abc223/tasks/abc223_h
[71] https://codeforces.com/contest/1336/problem/E1
[72] https://codeforces.com/blog/entry/68953
[73] https://codeforces.com/blog/entry/60003
[74] https://drive.google.com/drive/folders/1Ll8EuA3p64JLmzImfQu5qiWvc6_QTa0E?usp=sharing
[75] https://usaco.guide/adv/xor-basis?lang=cpp

Phỏng vấn Vũ Hoàng Kiên

Interviewer: Phạm Tuấn Nghĩa – ICPC World Finalist 2020 & 2022



Kiên: Xin chào tất cả mọi người, mình là Kiên. Hiện tại mình đang học năm 2 của trường Đại học Quốc gia Singapore. Rất vui được gặp các bạn!

Nghĩa: Kiên là người mới học CP[76] năm cấp 3 nhưng khi lên 11 đã đạt được HCB IOI[77] . Vậy Kiên đã làm gì để thăng tiến nhanh như vậy? Thói quen train CP của bạn có gì đặc biệt không?

Kiên: Nói như nào nhỉ, thật ra mình cũng không biết người khác train như thế nào nên mình cũng không biết việc train của mình có đặc biệt không. Về căn bản là mình sẽ học theo cảm hứng. Ví dụ thầy gợi ý 1 set bài để làm thì nhiều khi mình không cố làm đâu mà lúc nào hứng lên mới làm. Càng về sau thì mình thấy có chủ đề này chủ đề nọ thú vị nên mình càng năng suất làm bài hơn.

Nghĩa: Điều đấy có giải thích cho vì sao lớp 10 bạn chưa nổi mà đến lớp 11 bạn mới nổi không nhỉ? Bởi vì ban đầu bạn chưa train hardcore mà nó kiểu cảm hứng nhỉ?

Kiên: Lớp 10, thời gian đấy mình cũng khá là quằn. Mình cũng không được đi thi Duyên hải các thứ vì thời gian chọn đội Duyên hải nó cũng sớm hơn cái thời gian tryhard cho tụi lớp 10 trường mình. Sau khoảng tryhard vào tầm hè, mình cũng học thêm khá nhiều thứ. Rồi mình đỗ tuyển, trong thời gian đấy mình cũng tham gia chung mấy cuộc thi với tuyển. Rồi sau đấy thi QG rồi được Nhất QG thì hmm..., lúc đấy mình kiểu suýt được vô địch VOI, bằng rank anh Nghĩa . Hồi đấy mình bắt đầu thấy ảo rồi! Mình có đăng ký thi Quốc Gia bên Singapore nữa để trải nghiệm thử ấy. Nhưng thời gian học Vòng 2 nó quằn quá nên thi hơi thọt, còn bị mấy đứa không đỗ Vòng 2 hành. Sau vụ đấy mình cay, bắt đầu tryhard các thứ, thi Vòng 2 lại xanh!

Nghĩa: Okay tóm lại là bạn có thói quen train theo cảm hứng và train vào những lúc bạn cảm thấy mình ngu, những lúc mà bạn cảm thấy kém?

Kiên: Đúng rồi, hầu hết là như vậy ạ

Nghĩa: Ờm thì có vẻ cái quá trình học CP của bạn nó khá giống cái đồ thị hình sin. Khi nào bạn khủng thì bạn thọt dần rồi bạn lại bắt đầu khủng . Thế thì, giữa các cái lúc phong độ của bạn đi xuống ấy, lúc đấy bạn có chơi game, lướt Internet đọc reddit hay bạn đã làm gì khác mà phong độ của bạn đi xuống nhiều như vậy?

Kiên: Không biết lúc đấy có mấy cái Reels như giờ không, kiểu mình hay xem mấy cái giống vậy, mấy clip trên Youtube các thứ. Thực ra mình nghĩ Tin học là một môn cần luyện tập liên tục ấy, chỉ cần mình không làm gì thì cảm giác nó tự đi xuống thôi. Có thể tư duy nó vẫn thế nhưng cái cài đặt cũng đi xuống, đại loại vậy.

Nghĩa: Okay, rất là hợp lý. Nhưng mà, thường khi người ta không làm gì thì nó vẫn phải đi kèm với một cái hoạt động cụ thể nào đấy. Những lúc bạn không train hoặc train ít ấy, bạn có sở thích nào đấy ngoài CP không? Vì thực ra mình nghĩ có một cái gì đấy ngoài CP cũng rất là tốt. Nó cũng train cho mình cách nhìn các thứ theo nhiều góc khác nhau.

Kiên: Thật ra mình cũng có tham gia các cuộc thi kiểu liên quan đến thể thao mà kiểu chơi vui thôi chứ cũng không đến mức tryhard. Mình chơi game cũng leo lên được top bao nhiêu phần trăm của game đấy nhưng cũng không đầu tư nhiều quá vào game. Nói là top bao nhiêu phần trăm thì thế nhưng mà mình cũng không nhắm cao quá ấy. Kiểu cũng cho vui thôi!

Nghĩa: Thế là Kiên cũng có chơi game và bạn cũng có quan tâm đến cái thứ hạng của mình đang nằm trong phần nào của game. Thật ra bạn cũng ganh đua trong các hoạt động giải trí nhỉ? Mình cũng rất là đồng cảm tại mình cũng là một người như vậy. Bạn có thể kể thêm về hành trình của bạn đến NUS không? Quá trình app NUS của bạn có gặp khó khăn gì, lúc biết mình đỗ thì bạn có bất ngờ không?

Kiên: Mình có lợi thế về thi quốc tế này. Thật ra NUS cũng là một trường có uy tín trong khu vực về CP nên cái hồ sơ của mình cũng cho mình một số lợi thế. Thế nên là việc đỗ cũng không bất ngờ lắm. Mình cũng gặp khó khăn trong việc xin học bổng các thứ ấy, mình nghĩ là cái hồ sơ của mình nó không thuyết phục lắm. Hồi đấy NUS có một học bổng khác gần giống học bổng mình đang có bây giờ, nhưng không bắt phải ở lại Singapore 3 năm. Một số bạn mà mình quen đã đạt được cái học bổng xịn hơn đấy, và tất nhiên họ sẽ từ chối cái học bổng mình đang có. Bằng một cách nào đó mà tự nhiên cái học bổng này được dư thêm một số suất và mình nghĩ đấy là một điều may mắn.

Nghĩa: Về việc làm hồ sơ đi du học thì trước đây mình cũng có làm và với mình đó là một trải nghiệm rất là khác. Không biết là với bạn thì bạn có như thế không? Kiểu mình học được một cái kĩ năng gì đấy từ quá trình chuẩn bị hồ sơ này. Mình cũng biết rất nhiều bạn cũng train CP từ cấp ba, cũng đang trong quá trình làm hồ sơ du học. Không biết bạn có góp ý hay lời khuyên, chia sẻ

[76] CP: Competitive Programming - Lập trình thi đấu

[77] IOI: Kỳ thi Olympic Tin học Quốc tế.

tips gì cho các bạn để xin được học bổng khủng như bạn không?

Kiên: Về tips thì mình cũng không rõ vì mình không nghĩ mình là một người giỏi viết luận hay giỏi phỏng vấn các thứ. Nhưng mình nghĩ app du học là một cơ hội tốt để mọi người tự tìm hiểu về bản thân mình đấy. Tại vì mình hiểu mình thì mình mới quảng cáo bản thân mình đủ tốt để thuyết phục các trường nhận mình vào học. Mặc dù nó có thể hơi trầm cảm gì đấy nhưng ít nhất đó cũng là một tác dụng rất lớn của app du học.

Nghĩa: Với mình thì hồi mình nộp đi du học thì điều mình nhận ra đấy là cách chấp nhận giới hạn của bản thân và luận thì đưa người khác sửa. Chứ mình viết như dở hơi! Trong thời gian apply đi du học thì Kiên có nộp các trường khác không? Và nếu có thì tại sao Kiên lại chọn NUS thay vì các trường khác?

Kiên: Có, nhưng mà em không được chọn. Đơn giản thế thôi.

Nghĩa: Okay, rất ngắn gọn. Một câu hỏi cuối cùng về đời sống đại học. Việc bước lên Đại học nó đã là một sự thay đổi về cuộc sống hằng ngày với nhiều bạn, thêm cả sự khác biệt về văn hóa khá là lớn, ngoài ra bạn được học ở một ngôi trường Đại học nằm trong top rất cao của Châu Á và Thế giới. Thế bạn có thấy trong 2 năm vừa rồi, có những kĩ năng gì của bạn được cải thiện và có những kĩ năng nào mà vì ở trong môi trường như thế mà bị thụt lùi đi không?

Kiên: Nâng trình thì có thể mình vì chửi nhau bằng tiếng Anh nên nói và viết tốt hơn chăng? Còn về bị thụt lùi thì... Nói chung mình cũng là một người ngại giao tiếp với người lạ nên là một phần nào đấy mình hơi ít vận động từ hồi lên Đại học. Còn các kĩ năng liên quan đến học thuật thì nói chung mình để phần sau mình nói .

Nghĩa: Về quá trình học CP, cụ thể thì trong khoảng lớp 11-12 của bạn. Kiên có cảm thấy Kiên may mắn khi đã thăng tiến rất nhanh vào năm 11 hay không? Sang lớp 12 thì Kiên có còn động lực như trước để chinh phục HCV không hay chỉ đến đâu thì đến đấy thôi?

Kiên: Lúc nãy em có kể em thi thọt, rồi về sau cay ấy . Thật ra lúc đấy ban đầu em chỉ kiểu nhắm top 15 để lớp 12 chơi các thứ ấy ạ. Em chỉ nghĩ em tầm rank 8, 9 gì thôi, xong em được rank 3 nên em bắt đầu nghĩ mình cũng có khả năng đi quốc tế. Rồi lúc đấy em bắt đầu học khá căng, khoảng thời gian từ đấy đến lúc thi APIO[78] . Còn đoạn sau như nào nhỉ? Vẫn thế, không thay đổi nhiều lắm, cứ tryhard thôi!

Kiên: Lên lớp 12 thì có một khoảng thời gian em không tryhard được như trước để tập trung học những nội dung khác. Tức là em vẫn cố để được huy chương các thứ, nhưng có một khoảng thời gian em tạm thời dừng lại để tập trung làm cái khác, vì em cũng không phải thi vòng 1. Sau đấy thì em bắt đầu tryhard trở lại. Nói chung em cũng nhắm khá cao. Đến cuối thì mọi thứ của em vẫn không đổi, từ rank Vòng 2 đến huy chương này kia luôn . Nói như thế chứ em cũng chả biết kể chi tiết như nào. Có một cái em để ý là thi IOI thì ngày 1 em lúc nào cũng Vàng và ngày 2 lúc nào cũng Đồng. Xong là lấy trung bình ra là được Bạc, cái đấy em nghĩ khá đúng. Cả cái đồ thị hình sin anh có nhắc ở phần trước

Nghĩa: Chứng tỏ là có một cái "quy luật" về việc học của bạn xuất hiện khá rõ . Nếu vẽ cái biểu đồ ra, bạn có thấy mấy tháng gần đây nó còn đúng không? Trong việc bạn đi WF[79] , giành Medal đầu tiên cho Việt Nam và việc tiếp theo đấy là vào WF lần nữa?

Kiên: Có có! Em có thọt National .

Nghĩa: Được biết bạn là thủ khoa đầu vào của Chuyên KHTN khóa 2k2. Bạn có điều gì có thể chia sẻ cho các bạn học sinh cấp 2 về kinh nghiệm của mình không?

Kiên: Em cũng không nhớ rõ, hồi đấy lịch học của em là học tất cả mọi ngày trừ ngày trước ngày thi. Thật ra em cũng may mắn được tạo điều kiện khá nhiều, được ôn luyện ở nhiều chỗ khá uy tín ở trong quận.

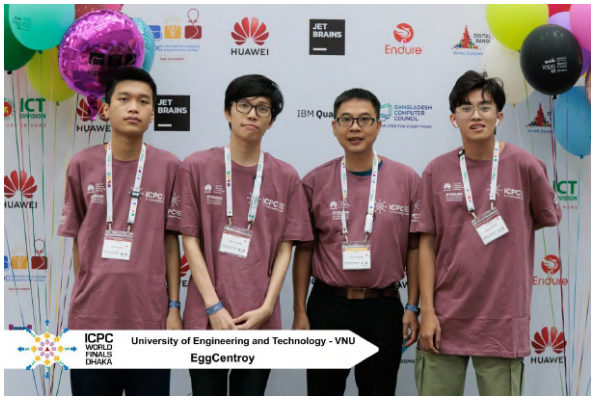
Nghĩa: Okay thế lời khuyên cho các bạn là hãy học thật nhiều và học đúng chỗ đúng không? Câu hỏi cuối cùng này, đã có khi nào Kiên nghĩ về tấm medal ICPC[80] WF từ thời điểm mình học cấp 3 không?

Kiên: Thực ra mình thi ICPC từ cấp 3 nhưng năm nào cũng bị hành nên là chưa bao giờ có giải ICPC khối THPT. Nên mình cũng chưa nghĩ đến.

[78] APIO: Kỳ thi Olympic Tin học Châu Á -– Thái Bình Dương.

[79] WF: World Finals – Cuộc thi ICPC cuối cùng trong một mùa giải bao gồm những đội xuất sắc nhất của các trường đại học trên thế giới vượt qua vòng loại vùng (Regional contest)

[80] ICPC: International Collegiate Programming Contest - Cuộc thi Lập trình Quốc tế lâu đời và danh giá nhất dành cho sinh viên các trường đại học và cao đẳng trên toàn cầu.



Vũ Hoàng Kiên tại kỳ thi ICPC World Finals 2021, Dhaka (Bùi Hồng Đức, Vũ Hoàng Kiên, thầy Hồ Đắc Phương, Nguyễn Hải Bình)

Nghĩa: Okay, sau khi nghe câu chuyện của bạn trong quá trình train ở cấp 3, mình thấy nó luôn luôn có một cái đích ở trước mắt nhưng mà luôn là đến giữa đường rồi bạn mới nhận ra mình thọt quá và bạn mới leo lên chứ cũng chưa bao giờ thấy bạn nhìn đến một cái goal xa hơn. Thực ra như thế thì nó chỉ phù hợp với một số bạn chứ không áp dụng được với nhiều người. Nước đến chân mới nhảy thì không phải ai cũng nhảy cao được như Kiên . Các bạn có thể thấy là trong một khoảng thời gian rất là ngắn, chỉ cần cảm thấy cần học thì Kiên sẽ vô địch, chấm hết!

Nghĩa: À mình cũng biết là dù Chuyên KHTN là một môi trường rất cạnh tranh, nhưng không phải khóa nào cũng có độ chênh lệch lớn về kĩ năng ngay từ khi bắt đầu vào như khóa bạn, kiểu Bùi Hồng Đức và phần còn lại. Theo bạn thì việc có một người bạn rất là mạnh ngay từ điểm xuất phát rồi thì bạn có thấy đấy là động lực trong việc học CP không?

Kiên: Em lại nghĩ đấy là một điều tốt! Ít nhất mình luôn có một khuôn mẫu để nhìn theo. Đại loại nếu có người như thế thì cũng khá tốt trong việc giữ động lực và phát triển bản thân.

Nghĩa: Thế lúc mà bạn train thì mình có nhìn những bạn xung quanh bạn không? Hay bạn chỉ nhìn theo mỗi Hồng Đức thôi?

Kiên: Thực ra cái việc soi mói người khác thì không phải lúc nào cũng nên thế. Vì sống mà áp lực thế thì khó chịu lắm, cũng tùy lúc chứ ạ. Lúc thi cũng có nhiều cái lắm, mình cứ làm việc của mình trước thôi, khi nào xong rồi thì mình mới rảnh để nhìn linh tinh

Nghĩa: Đúng là chất của top 3. Bạn train vì cảm hứng, train vì đam mê chứ không train vì mục tiêu gì cả . Thực ra thì mình cũng biết điều này ngay từ đầu nên mình cũng rất là khó hỏi, cách học của bạn cũng khó để áp dụng một cách phổ thông cho mọi người. Thế nên hỏi lời khuyên đành cho các bạn khác thì cũng hơi khó.

Nghĩa: Một câu hỏi cuối cùng của BTC. Sau khi nhận được medal của WF ở năm nay thì bạn còn tiếc nuối gì về CP không? Có điều gì bạn cảm thấy mình còn cần phải cố gắng để đạt được ở CP không?

Kiên: Thật ra nói có thì cũng có. Mình cũng từng khá gần với Legendary Grandmaster (rating 3000) ở Codeforces[81] ấy, nhưng sau 1 đợt mình không tập trung nữa thì mình xuống khá là sâu. Nên là ít nhất mình hy vọng mình có thể chạm đến cái mốc đấy trước kì WF cuối cùng của mình.

Nghĩa: Okay chúc bạn may mắn nhé! Vì mình cũng thấy cũng lâu rồi cái mốc rating nó không đổi nên mình nghĩ từ giờ đến lúc đấy có khi là... Thôi thì chúc bạn là Codeforces sẽ không tăng cái nấc LGM lên, nếu không thì sẽ hơi mệt mỏi đấy!

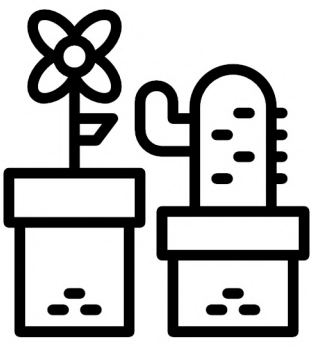
Nghĩa: Okay, chắc là nó chỉ vậy thôi. Việc bạn top 3 đã khó hỏi rồi mà cách train của bạn cũng khá đặc biệt nữa. Mình nghĩ nếu mà mình đu theo cách train của bạn thì mình cũng khó khuyên cho các bạn khác . Thế thôi, cảm ơn bạn nhé. Chào bạn!

Kiên: Chào anh Nghĩa ạ .

[81] Codeforces: một trang web tổ chức các cuộc thi lập trình thi đấu với các dạng đề, bài tập đa dạng.

Phỏng vấn Trần Xuân Bách

Interviewer: Nguyễn Trung Quân – Tình nguyện viên VNOI Gen 2



Quân: Chào Bách, bạn có thể giới thiệu đôi chút về bản thân cũng như thành tích nổi bật của bạn được không?

Bách: Em tên là Trần Xuân Bách, học lớp 12 trường THPT Chuyên Khoa học Tự nhiên, Hà Nội. Năm lớp 10, em đạt giải Nhì VOI 2020, huy chương đồng APIO[82] . Sang năm lớp 11, em may mắn có cho mình huy chương vàng APIO và huy chương vàng IOI[83] .

Quân: Được biết Bách bắt đầu học Tin từ rất sớm, không biết cơ duyên nào đã đưa bạn đến môn học này?

Bách: Hè năm lớp 6, mẹ có tìm cho em một trại hè. Đó là HSGS Codecamp của thầy Phương tổ chức. Vào đấy, em được học về C++, được làm trang codefun của thầy Phương. Em nhận ra việc code khá logic đối với một người chuyên toán như em. Và sau khi biết được cách những dòng code hoạt động, em đã tự mày mò khám phá, từ đó bắt đầu tập trung sâu hơn về môn này.

Quân: Cảm xúc của bạn như nào khi biết được mình được tuyển thẳng vào THPT Chuyên KHTN?

Bách: Em được biết nếu được giải Nhất thì sẽ được tuyển thẳng vào chuyên KHTN từ năm lớp 8 rồi, em cũng đã thi từ năm lớp 8 nhưng chỉ được giải Nhì thôi. Lúc biết mình đạt giải Nhất TP, em cảm thấy khá là may mắn và cũng tiện nữa .

Quân: Đối với Bách thì việc học ở cấp 3 và cấp 2 có khác biệt gì quá lớn không?

Bách: Thật ra thì em học trên lớp rất ít. Em chưa lên lớp buổi nào bao giờ luôn, toàn học đội tuyển thôi . Theo em thấy ở KHTN thì sự khác biệt lớn nhất nằm ở sự tự giác của học sinh. Ở trường Ngôi Sao, thầy cô hay ép tụi em làm đề cương, sách ôn thi các thứ, còn lên cấp 3 thì chắc là kiểu trưởng thành hơn nên em thấy mọi người có tự giác tự học, làm bài tập rồi còn tự nghiên cứu thêm nữa. Đặc biệt là môn Tin thì phần tự nghiên cứu ấy theo em là quan trọng nhất.

Quân: Bạn xem việc tự học là một trở ngại hay là thuận lợi trong con đường lập trình thi đấu của mình?

Bách: Em tự cảm thấy mình tự học khá kém, do dễ bị mất tập trung. Nhưng mà không hiểu sao em vẫn đủ giỏi, chắc là mấy bạn khác mất tập trung hơn em . Em nghĩ đấy là một bước đệm khá tốt để sau này vào Đại học, mình sẽ được chuẩn bị trước tinh thần tự học ấy ạ.

Quân: Trải qua quãng thời gian học đội tuyển cũng lâu thì động lực để Bách duy trì việc học liên tục là gì?

Bách: Thật ra em cũng không rõ. Chắc chính áp lực khi học đội tuyển là động lực tốt nhất đối với em. Thầy có nói với bọn em rằng: "Anh không tiến, thì anh sẽ lùi." Thế nên kể cả khi không muốn học thì em vẫn phải ép mình học để không bị đánh mất kiến thức, không bị thụt lùi so với các bạn cùng lứa. Thế cho nên em nghĩ rằng có khả năng tự học là một lợi thế khá lớn

Quân: Theo mình được biết rất hiếm trường hợp lớp 10 được vào đội tuyển của trường KHTN, khi Bách học tập trong môi trường với nhiều người khủng thì có áp lực không?

Bách: Em có được học cùng đội dự tuyển của Tự nhiên từ hồi cấp 2 nên những cái đấy chắc em cũng trải nghiệm quen rồi, cũng 3-4 năm rồi ạ. Bởi thế em cũng không thấy áp lực lắm, dù em biết mình có thể không bằng một số người nhưng xét về mặt bằng chung thì em cũng ổn. Kỳ thi tuyển lần ấy em đứng thứ 2, chỉ thua một anh lớp 12 thôi .

Quân: Trước khi vào đội tuyển, Bách đã có nhiều trải nghiệm khi học dự tuyển của trường KHTN, việc học "kế" như thế là trải nghiệm như thế nào đối với bạn?

Bách: Thật ra đợt học dự tuyển đấy em cũng khá là bị áp lực, nhiều lúc có một số anh trong đội học bật hẳn lên làm em có cảm giác bị thụt lùi ý. Em cũng thấy căng thẳng, nhưng khi nhìn vào mặt khác thì em cũng biết mình nhỏ tuổi hơn nên vẫn còn nhiều cơ hội và lợi thế về sau này, kiểu so với khóa sau thì có thể em vẫn hơn được chẳng hạn .

Bách: Lúc đấy được học cùng các anh lớn hơn mình tận 3 lớp, chắc là em cũng thấy sợ . Bước vào lớp nhìn ai cũng cao hơn mình nửa mét cơ mà, kiểu là một trải nghiệm mới hoàn toàn đối với em, khi mà vừa vào cấp 2 hơn 1 năm, xong tự dưng được trải nghiệm môi trường cấp 3. Đối với em đó là một cú sốc khá là lớn, em chưa từng trải nghiệm cái cảm giác học mà "buông thả" như thế, trong giờ học mình có thể không học cũng được và thầy cũng không nhắc nhiều. Thời gian đầu khi học như thế em cũng "nghiện ngập" khá nhiều , qua thời gian thì em cảm thấy dần hứng thú với việc học hơn nên cũng bỏ dần, cho nên em thấy là khi mình được tự do như thế thì bản thân cũng phải biết kiềm chế nhiều hơn.

Quân: Ở kì IOI vừa rồi Bách đã thể hiện rất tốt và đem về tấm huy chương duy nhất cho đoàn Việt Nam, Bách có thể kể hành trình Bách đến với

[82]APIO: Kỳ thi Olympic Tin học Châu Á -– Thái Bình Dương.

[83]IOI: Kỳ thi Olympic Tin học Quốc tế.

kì thi IOI được không?

Bách: Phải bắt đầu từ năm lớp 10, lúc đấy việc vào đội tuyển cũng không bất ngờ đối với em vì trước em cũng có biết anh Bùi Hồng Đức và anh Lê Quang Huy làm được tương tự . Lúc có kết quả thì em được hạng 2 toàn tuyển, dẫu là lúc đi thi em cũng chỉ cắn test thôi nên không tự hào nhiều nhưng ít nhất cũng cho em sự tự tin . Sau đó em thi VOI và được giải Nhì, tí nữa em trượt Vòng 2 vì rank của em đợt đấy tầm nửa dưới của top 32 cơ. Khi thi xong em cũng không có hối hận nhiều vì em hầu hết dành được điểm trọn vẹn những subtask mà em làm được. Hồi đấy nghĩ lại em còn khá yếu về một số mặt, như kiểu Quy hoạch động những bài trên Codeforces[84] em còn không làm được những bài rating 1900, em không hiểu sao nữa . Thế nên là sau cuộc thi em cũng học nhiều hơn và cũng dần khắc phục được những hạn chế ấy.

Bách: Ở Vòng 2 thì em cũng chỉ biết cắn test cả 6 bài. Duy chỉ có một bài em cắn mà sai đúng 1 test, không là em 100 điểm . Tổng quan thì ngày 1 em top tầm 25, ngày 2 em bật lên top 11. Khoảnh khắc đấy thì em biết khả năng mình vào IOI khó rồi, dù thầy từng động viên có trường hợp top 11 thì vẫn vào được nhưng em vẫn không tự tin lắm nên em cũng chỉ cố thi APIO thôi . Thi APIO thì, ui giời . Năm ấy Indonesia tổ chức, thi 5 tiếng thì sever bị sập trong 3 tiếng cuối. 2 tiếng đầu em nộp thì ok, 3 tiếng cuối như thi VOI ấy ạ, em toàn gọi contest đấy là VOI+ . Như em có kể ở phần trước thì thi VOI em thường không mất điểm nào nên đến lúc em biết sever sập thì em chuyển hẳn thái độ thi của mình luôn, ngồi stress test đủ thứ. Cơ mà hóa ra chính cái đấy lại giúp em vào top 6, như anh Dương Minh Khôi chẳng hạn, nếu năm ấy server không bị tình huống như thế thì cũng có thể AC bài 2 và bài 3, mà đen là anh ấy code 2 bài đấy trong lúc server sập cơ mà lại không stress test nên không bắt được những bug khá là dễ nên thành ra mất điểm. Theo em thì năm đấy em cũng khá rùa, tự nhiên lại được thôi .

Bách: Trong cùng năm, thì em cũng có thi ICPC[85] nhưng kết quả của team em không tốt lắm do đợt đấy team em cũng không có ai code tốt cả, đợt đấy em code cũng vừa vừa thôi chứ không giỏi . Đến năm lớp 11 thì do em được thi APIO rồi nên không phải thi VOI nữa mà chỉ cần tập trung thi Vòng 2 thôi, cho nên em cũng có nhiều thời gian để thi ICPC, team của em năm đấy có Hùng với Khuê. Mỗi người có một thế mạnh riêng nên bọn em kiểu bù trừ cho nhau ý, cuối cùng thi thì kết quả cũng khá ổn .

Bách: Sau thi ICPC thì em thi Vòng 2, ngày 1 top 13-14, ngày 2 em ăn may nên bật lên được top 3 . Với kết quả đấy em được tham gia APIO, 2 tiếng đầu em đã ăn được trọn điểm em có thể ăn rồi, còn lại chả biết làm gì nữa, xong kiểu em cứ sợ nên cứ cố nghĩ tiếp, cuối cùng thì ra em cũng ra giải cho mấy bài đấy nhưng chả code được bài nào, có một bài mà mọi người đều được 90 rồi, và ít ai đẩy nó lên 100, với thời gian ít ỏi đấy em cũng chỉ kịp cắn test rồi ăn được thêm 2,9 điểm. Kể cũng hay vì nếu em không ăn điểm lẻ đấy thì VN mình đã có 5 Vàng APIO rồi, rồi cuối cùng em ăn điểm đấy nên em được bét Vàng còn anh Nhật Minh thì top Bạc, khổ thân anh Minh . Thật ra trong lịch sử Việt Nam cũng nhiều người bét Vàng, như có anh Cao Nguyên chẳng hạn, nhưng chắc em là người đầu tiên bét Vàng cả 2 lần trong một năm. Thi IOI thì em thấy mình cũng làm ổn. Ngày 1 ok, ngày 2 cũng ok nốt. Nói chung em làm cũng bình thường thôi. Trong đấy em AC cả 2 bài đều khá muộn, tầm 4 tiếng 40 phút. Em cũng thấy mình khá là may mắn khi không gặp bug nặng mà chiến thuật cũng không sai. Em không dính bug và vẫn có cái để code, gần như là 5 tiếng liên tục em code cả luôn. Kể cả em không AC nhưng em vẫn cắn được hầu hết subtask ở ngày 1, cũng nhờ việc cắn subtask này nên em cũng có hướng để AC được bài 1 ngày 2. Em nghĩ đấy là một sự may mắn khi tổng kết ra cả 2 ngày thì em lại được Vàng .

Quân: Vậy bạn đã chuẩn bị những gì cho kì thi Vòng 2 sắp tới? Không biết năm nay Bách có đặt cho mình những mục tiêu nào không, như được 2 Vàng IOI chẳng hạn?

Bách: Em vẫn sẽ ôn thôi. Việc ôn nhiều hơn thì em nghĩ là không, mà cũng không ít hơn năm trước là mấy. Vì em nghĩ cũng năm cuối rồi nên phải chuẩn bị cho Đại học nữa. Mục tiêu chung thì em nghĩ em cứ vào IOI đã rồi tính tiếp, còn việc được 2 vàng IOI thì em cũng muốn nhưng đạt được hay không thì em cũng không chắc, được như anh Bùi Hồng Đức thì cũng khó mà

Quân: Cộng đồng VNOI đã đóng vai trò như thế nào trong con đường học tập của Bách? Theo bạn thì việc tìm thấy những người bạn có cùng đam mê, cùng nhau phấn đấu trong con đường Tin học

[84] Codeforces: một trang web tổ chức các cuộc thi lập trình thi đấu với các dạng đề, bài tập đa dạng.

[85] ICPC: International Collegiate Programming Contest - Cuộc thi Lập trình Quốc tế lâu đời và danh giá nhất dành cho sinh viên các trường đại học và cao đẳng trên toàn cầu.

có quan trọng không?

Bách: Cộng đồng VNOI là nơi mà em biết có những người khác vẫn đang học giống như mình, và sẽ sẵn sàng giúp mình mỗi khi mà mình cần. Em vẫn nhớ hồi đầu VNOI mới lập Discord cho đến giờ, vẫn đang có rất nhiều admin, cũ mới, các bạn thường xuyên active để giúp cộng đồng càng ngày càng lớn dần. Đối với em VNOI là một cộng đồng mà mọi người có thể nói chuyện thoải mái với nhau về môn Tin và cùng mindset giống nhau ý. Em cảm thấy được hỗ trợ khá nhiều. Dù rằng em đã có các bạn đội tuyển ở trường, học cùng nhau rất lâu nhưng VNOI vẫn là một cái rất riêng. Với sự mở rộng của VNOJ rồi Wiki thì em nghĩ cộng đồng Tin học Việt Nam sẽ đi lên thôi

Quân: Vì sao bạn quyết định tham gia Kỳ thi VNOI Cup? Bạn cảm nhận như thế nào về phần thể hiện của mình ở cuộc thi này?

Bách: Nếu mà nói không vì giải thưởng thì em tự lừa dối bản thân quá . Thật ra mấy contest trên VNOJ thì em vẫn tham gia thường xuyên. Em cứ đăng ký thi thôi nhưng tự nhiên cả đội tuyển IOI lúc đấy được mời thi Final VNOI cup ở Đà Nẵng, thế nên là kể cả em không đăng ký thi thì em vẫn được vào Đà Nẵng thi với mọi người . Lý do thì chắc mỗi thế thôi, kỳ thi cũng giúp em luyện tập này, không trùng với IOI, lại có tiền giải thì em tham gia thôi. Phần nữa là em cũng muốn được gặp mặt mọi người vì mình có offline ở Đà Nẵng ý.

Trần Xuân Bách tại kỳ thi VNOI Cup 2022

Bách: Phần thể hiện thì em tự thấy mình khá là tệ. Bài 1 em bug này, mấy bài kia em làm bình thường rồi cố đấm bài 5, ban đầu em tưởng ra được giải rồi những cuối cùng chỉ làm được một nửa thôi nên em bỏ đến cuối giờ. Sau đấy, em ngồi tập trung code bài Hình vì đấy là sở trường của em xong ra được code tối ưu. Làm xong bài đấy thì em ngồi code bài có dùng Lagrange, tầm bài thứ 6, rồi em dính bug đến cuối giờ luôn . Bài đấy là bài Interactive, em bị bug đầu tiên bởi vì cái input của em bị sai tầm 2 phút cuối trước khi hết giờ, em sợ cho nên là vội xóa hết mấy cái debug đi để nộp. Về sau em xóa nhầm một cái dấu '?' trong câu interact nên là bị sai, tại cái đấy nên em mất giải Nhất và tầm 10 triệu

Bách: Bản thân em nghĩ em có thể làm tốt hơn hoặc full đề được, nhưng mà em không . Nhưng mà thôi em nghĩ thế là được rồi bởi vì LQDOJ Cup có admin bên VNOI đứng nhất thì VNOI Cup cũng có admin bên LQDOJ đứng nhất, em nghĩ thế là đều rồi.

Quân: Nếu không chọn môn Tin học, Bách nghĩ mình của hiện tại sẽ thế nào?

Bách: Khả năng lớn nhất thì em sẽ học tiếp Toán, bởi vì đằng nào không học Tin thì môn Toán của em cũng khá ổn. Nếu em nghĩ lúc đấy em tiếp tục cố gắng thì có cơ hội đi IMO. Chắc là ngày xưa thấy Toán khô khan quá nên em bỏ mất hoặc có thể em sang Vật lý này, em nghĩ thế ạ.

Quân: So với những học sinh cấp 3 thông thường, Bách đã có những trải nghiệm rất khác, có bao giờ bạn cảm thấy lạc lõng hay muốn được trải nghiệm cuộc sống của một học sinh bình thường không?

Bách: Nói lạc lõng thì cũng không hẳn nhưng mà nói về khác biệt thì em nghĩ là có. Em gặp mặt các bạn trong lớp chưa đến 10 lần, em còn chẳng nhớ tên các bạn. Em gần như không có cấp 3, em chỉ có học Tin thôi. Nếu sau này họp mặt với các bạn cấp 2, nghe mọi người nói chuyện về cấp 3, thì em có thể em thấy lạc lõng, nhưng bây giờ thì không anh ạ, bởi vì em có các bạn trong đội tuyển này, mọi người vẫn vui vẻ nói chuyện với nhau bình thường, nó cũng như một cái lớp học rồi .

Quân: Là một người thường xuyên duy trì việc học, có bao giờ Bách cảm thấy chán học không?

Bách: Đương nhiên thì em sẽ có lúc chán rồi quay ra chơi game hoặc làm thứ khác. Những lúc như thế để quay lại việc học thì em nghĩ động lực lớn nhất chính là em sẽ nhìn lại cả một chặng đường dài, nếu em từ bỏ giữa chừng thì em phải học lại từ đầu những kiến thức trên lớp. Và dĩ nhiên em cũng không muốn trải nghiệm cảm giác đấy lắm nên em cứ phải cố thôi . Thật sự thì em nghĩ là để lấy lại cảm hứng học tập thì em chỉ có thể giải lao 1 tuần là tối đa rồi sau đấy có thể quay vào việc học liền luôn. Nhiều lúc em cũng được hỏi lời khuyên dành cho các bạn chán học Tin không, thật ra em cũng chịu, em chán suốt ấy mà . Đấy là vấn đề riêng của mỗi người nên em cũng không thể đưa ra câu trả lời chung chung được.

Quân: Vào những thời gian rảnh thì Bách thường làm gì ngoài việc học bài trên lớp?

Bách: Chắc em cũng vẫn làm bài trên lớp . Ở nhà, nếu em không siêng nữa thì em quay ra làm việc khác. Có một dạo em cứ chơi game suốt, có dạo thì em đọc đủ thứ truyện, bây giờ em vẫn đang đọc nhưng mà ít thôi. Đại khái em có thể buông hưng không buông hẳn vì em biết được là "nghiện nhưng nghiện ít thôi", mình phải kiềm chế bản thân! Em sợ mấy cái game như LOL vì em biết mình mà dính vào cái thì rứt ra khá khó .

Quân: Bạn có dự định, mục tiêu gì cho năm 2023? Đã và đang chuẩn bị được gì cho ĐH rồi? Sau khi lên ĐH thì Bách có dự định CP[86] tiếp hay sẽ tập trung cho việc trau dồi kỹ năng, kiến thức khác?

Bách: Năm 2023 chủ yếu em sẽ cố để làm hồ sơ để đi du học, chắc là em sẽ học 1 năm tại UET, trong lúc đấy em sẽ dành thời gian để làm việc khác, ví dụ như là dev bình thường, làm web hay làm game. Em cũng đang có 1-2 dự án, chắc là sẽ dành thời gian cho mấy cái đấy. Về chuẩn bị, so với mọi người em thấy còn khá ít, mai em thi IELTS , em thi SAT xong rồi. Đại khái là em thi thố xong rồi, em đang nghiên cứu về các trường và sắp tới thì sẽ chuẩn bị viết luận. Thật ra nó cũng chung chung thôi, em cũng không rõ vào đại học sẽ có những gì nhưng em hy vọng em sẽ qua được hết . Với lại em nghĩ lên đại học em sẽ tập trung vào cái khác chứ không phải CP. Năm đầu em học UET thì em có thể thi ICPC, cố đến WF[87] này. Nhưng mà sau đấy thì khó, em nghĩ mình cũng không theo CP đến hết đời được, và ICPC cũng là cái cuối cùng rồi.

Quân: Bản thân Bách là người code rất chắc và ít bug, theo Bách đâu là phương pháp để Bách code được như thế?

Bách: Em nghĩ cái cứu em nhiều nhất thì vẫn chỉ là cái phần sinh test thôi, cái đấy có được nói ở trong một bài VNOI Wiki ấy. Bình thường, em cứ lôi bài Codeforces ra, mọi bài cố sinh test để cố gắng nộp 1 lần. Trong lúc train thì em cũng train như kiểu VOI luôn, chỉ cố gắng nộp 1 lần thôi. Em nghĩ đấy là một điều tốt, mọi bài luôn chứ không chỉ riêng bài thầy cho thì em mới làm như thế. Nếu mình không làm được đủ quen thì đến lúc thi VOI thì sẽ không làm nhanh những thao tác ấy được. Vì mình chỉ có 3 bài trong 3 tiếng thôi. Ngoài ra em trong lúc train em có cố gắng code không compile nữa, trải nghiệm này nó kiểu... ác mộng! Nhưng mà cái đấy giúp em được khá nhiều thứ, em đã code tầm 50-100 bài mà không compile rồi . Nếu như mình làm cái đấy quen, khi code thì mình phải làm đúng đến từng dấu chấm phẩy, dấu ngoặc nên mấy cái việc khác như kiểm tra kiểu dữ liệu biến, kích cỡ mảng nó cũng sẽ đến một cách tự động luôn.

Quân: Còn không bao lâu nữa thì kì thi HSGQG sẽ diễn ra, bạn có bí quyết gì muốn chia sẻ gì với các bạn học sinh sắp tham gia không?

Bách: Em nghĩ độ khó của VOI thì mình làm ổn các bài độ khó tầm 2100 trên Codeforces. Tức là rating Codeforces của anh tầm phải 2100-2200, làm được các bài đấy ổn định trong vòng 1 tiếng thì thi VOI sẽ đảm bảo trong top 32 luôn, nhưng mà thật ra cũng còn tùy năm nữa. Em có thi một lần và xem mọi người làm bài một lần nhưng mà cả 2 năm nó đều kiểu gì ý, nhất là năm ngoái . Năm của em thì tất cả các bài đều ở mức trung bình, nên em cũng chỉ khuyên được thế thôi.

Quân: Nếu có cơ hội thi VOI lại một lần nữa thì bạn có tự tin mình sẽ được giải Nhất Quốc gia không?

Bách: Em nghĩ chắc là được ạ, nhưng Nhất top mấy thì em không rõ .

Quân: Trong tin thường có những thuật toán rất khủng, Bách có thường xuyên dành thời gian để học những thuật như thế không?

Bách: Cái phần kiến thức mới thì em chưa học trong vòng tầm nửa năm gần đây rồi. Nói sao nhỉ? Học những cái đấy thì đối với em cũng chỉ để cho vui thôi chứ cũng không áp dụng được IOI, mà nếu có áp dụng được thì chắc 99% người khác cũng không biết làm ngoại trừ 4 ông Trung Quốc đâu nên em cũng thấy yên tâm .

Theo em thấy, kiến thức của bên Tin khá ít so với Toán, đặc biệt là nội dung để thi IMO. Em thấy Toán là môn duy nhất mà kiến thức không bao giờ có giới hạn. Còn bên Tin thì đến tầm rating 2400 trên Codeforces là gần như hết mọi thứ để mình học rồi.

Quân: Trong ngày bạn dành bao nhiêu thời gian cho việc học thuật toán và code, ngoài thời gian học thì bạn thường làm gì?

Bách: Thời gian code đỉnh điểm là lúc train IOI em dành tầm 10 tiếng mỗi ngày và 8-9 tiếng ngủ. Phần còn lại em dành để làm mấy hoạt động hàng ngày như ăn uống rồi di chuyển các thứ .

[86] CP: Competitive Programming - Lập trình thi đấu

[87] WF: World Finals – Cuộc thi ICPC cuối cùng trong một mùa giải bao gồm những đội xuất sắc nhất của các trường đại học trên thế giới vượt qua vòng loại vùng (Regional contest)

Quân: Bạn thường dành ra bao lâu để nghĩ một bài khó và sau bao lâu không nghĩ ra thì đọc lời giải?

Bách: Em nghĩ là tầm 2 đến 3 tiếng. Thường thì em train theo kiểu em sẽ đọc 3, 4 bài cùng một lúc, xong rồi em ngồi nghĩ 3 đến 4 tiếng, nếu mà không nghĩ ra thì em đi đọc lời giải. Chia thời gian dành cho mỗi bài thì nó hơi khó nhưng mà chắc tầm 3 tiếng, thật ra 3 tiếng cũng hợp lí vì nếu nghĩ bài trong 3 tiếng mà không ra được 1 bài thì lúc đấy mình cũng hết thời gian của VOI, hay IOI thì cũng mất nửa ngày thi rồi. Thói quen này thật ra cũng xuất phát từ việc học đội tuyển của bọn em thôi, bọn em thường có 3-4 bài rồi ngồi làm trong 1 buổi sáng hoặc 1 buổi chiều, xong sau buổi học mọi người có thể tiếp tục nghĩ hoặc đọc lời giải luôn, em nghĩ cái đấy là tùy mỗi người, riêng em thì em thường đọc lời giải luôn.

Vô tình thì lúc nãy em cũng đọc một cái blog khá là hay. Ý tưởng của nó là mình nên học theo cảm xúc. Em nghĩ cái cảm xúc của mỗi người nó quyết định khá nhiều, kiểu như là trực giác bản thân ấy. Nếu mà mình nghĩ 1 bài 5 tiếng mà vẫn thấy vui vẻ, vẫn tìm tiếp được hướng thì mình cứ nghĩ đi. Còn nếu mình stuck 1 bài mà chỉ trong 1 tiếng mình đã không nghĩ ra gì nữa rồi thì đọc lời giải luôn cũng được, thế nó hợp lý hơn. Nói chung là khi nào em cảm thấy chán thì em đọc thôi .

Quân: Sau Tết sẽ diễn ra kỳ thi Học sinh giỏi Quốc gia, Bách có lời khuyên về động lực để gửi đến các bạn trong kỳ thi sắp tới không?

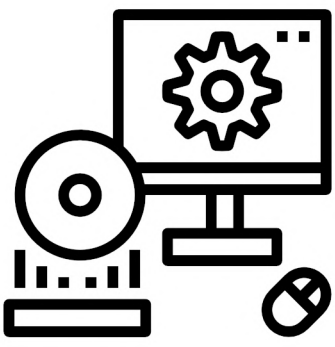
Bách: Động lực để vượt qua thì em cũng không biết nói sao nữa . Trong giờ thi mà nghĩ cái đấy thì em nghĩ mình đã thất bại một phần rồi, kiểu như tự tạo áp lực cho mình ấy. Bình thường trước giờ thi em cứ làm mấy cái như hít thở sâu cho bình tĩnh lại này, hoặc thầy em hay bảo em ngồi thiền, em thấy cái đấy cũng khá tốt. Cái đấy có thể tùy ở mỗi người thôi nhưng mà em nghĩ cái quan trọng nhất chính là lúc mình làm bài train thì cố gắng tạo không gian giống phòng thi, nên đến lúc thi ấy em có cảm giác mình đang làm bài trên lớp thôi . Dĩ nhiên là mình phải cố gắng hết sức rồi, em cứ cố gắng code mọi lúc có thể, nghĩ mọi lúc có thể. Tóm lại là không được ngồi chơi và tạo cho bản thân cảm giác thư giãn. Trước lúc thi em cũng không nghĩ đến chuyện trượt và đậu, cứ tập trung cao độ vào bài làm thôi. Tại em nghĩ khi ở trong trạng thái tập trung rồi thì em cũng không nghĩ được điều khác nữa.

Quân: Cảm ơn Bách đã tham gia phỏng vấn và những chia sẻ rất thú vị của bạn! Chúc bạn một buổi tối vui vẻ!

Bách: Chào anh ạ!

Áp dụng bất ngờ của đạo hàm

Nguyễn Thành Trung
Singapore



Lúc học môn Toán có lẽ chúng ta thường hay tự hỏi: Học đạo hàm, tích phân.. để làm gì? Hôm nay chúng ta sẽ cùng nghiên cứu một áp dụng vô cùng bất ngờ và ảo ma của đạo hàm trong 1 bài toán.

Bài toán như sau:

- Cho N ngôi nhà trên 1 đường thẳng. Ngôi nhà thứ i có H_i tầng.
- Để khu phố nhìn đẹp hơn, người ta muốn sửa lại các ngôi nhà sao cho độ cao của tất cả các ngôi nhà đều bằng nhau, bằng cách xây thêm hoặc đập đi một số tầng của một số ngôi nhà.
- Với ngôi nhà thứ i, chi phí để xây thêm 1 tầng hoặc đập đi 1 tầng là C_i.
- Tìm chi phí nhỏ nhất.
- Giới hạn: $N \leq 10^7$, $0 \leq H(i) \leq 10^7$, $0 \leq C_i \leq 10^{11}$.

Ví dụ: với $N = 3$, $H = [10, 20, 15]$, $C = [1000, 1, 2]$. Cách tối ưu là đưa cả 3 nhà về độ cao 10, với chi phí:

$$|10-10| * 1000 + |20-10| * 1 + |15-10| * 2 = 20$$

Các bạn có thể đọc đề và nộp thử tại SPOJ [88] (tuy nhiên giới hạn trong bài SPOJ này khá nhỏ).

Lời giải $O(N * log)$

Đầu tiên chúng ta hãy nghiên cứu 1 số lời giải "thông thường" cho bài toán này.

Đặt $f_i(x)$ là chi phí để sửa ngôi nhà thứ i về độ cao x. Ta có công thức:

$$f_i(x) = |x - H_i| * C_i$$

Đặt $F(x) = \sum_{i=1..n} f_i(x)$, nói cách khác $F(x)$ là chi phí để đưa tất cả các ngôi nhà về độ cao x. Đề bài yêu cầu ta tìm $min(F(x))$.

Nhận xét 1:

Đặt x_0 là giá trị của x để hàm $F(x)$ đạt giá trị nhỏ nhất, khi đó:

$$min(H_i) \leq x_0 \leq max(H_i)$$

Khá hiển nhiên, giả sử $x_0 < min(H_i)$, điều này chỉ xảy ra khi ta phải giảm độ cao của tất cả các ngôi nhà đi ít nhất 1 tầng. Với mỗi ngôi nhà ta giảm nó ít đi 1 tầng thì sẽ được chi phí nhỏ hơn. Do đó $x_0 \geq min(H_i)$.

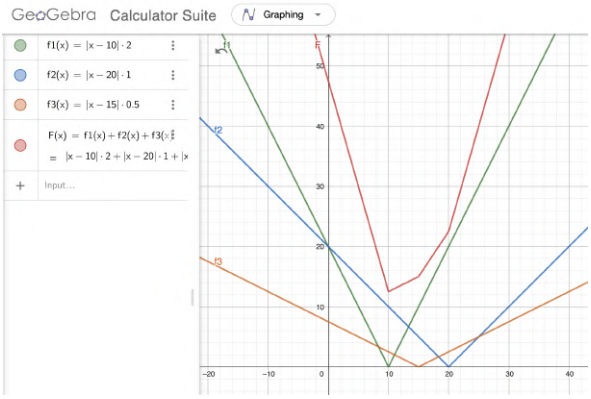
Tương tự, ta cũng chứng minh được $x_0 \leq max(H_i)$

Với nhận xét này, ta có thuật toán $O(N * max_H)$, với max_H là giá trị lớn nhất của H: duyệt tất cả các giá trị của x, với mỗi x, ta tính hàm $F(x)$ trong $O(N)$.

Nhận xét 2:

Hàm $F(x)$ là tổng của các hàm lồi (strictly convex) nên nó cũng là hàm lồi.

Ví dụ: trong ảnh mình vẽ hàm $F(x)$ màu đỏ là tổng của 3 hàm $f_1(x)$, $f_2(x)$ và $f_3(x)$. Có thể thấy $F(x)$ là hàm lồi



Do đó ta có thể sử dụng tìm kiếm tam phân [89] để tìm giá trị nhỏ nhất của $F(x)$.

$\Rightarrow$ đến đây ta đã có thuật toán với độ phức tạp $O(N * log(max_H))$.

Lời giải $O(N + max_H)$

Rất bất ngờ, bài này còn có thể giải được với độ phức tạp $O(N + max_H)$ sử dụng đạo hàm!

Ở bài viết này chúng ta ứng dụng đạo hàm của hàm trên số nguyên ($f(x)$ với x chỉ nhận giá trị nguyên) - discrete derivative [90] . Định nghĩa:

$$f'(x) = f(x) - f(x-1)$$

Áp dụng công thức trên vào bài toán ban đầu ta tính đạo hàm của hàm chi phí

$$f_i'(x) = \begin{cases} -C(i) & \text{nếu } x \leq H(i) \\ C(i) & \text{nếu } x > H(i) \end{cases}$$

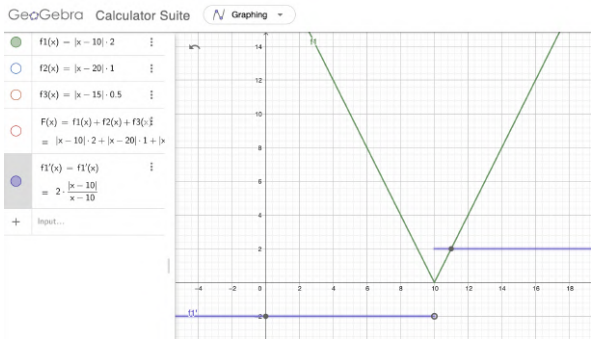
[88] https://www.spoj.com/problems/KOPC12A/
[89] https://vnoi.info/wiki/translate/emaxx/Tim-kiem-tam-phan-Ternary-Search.md
[90] https://calculus.subwiki.org/wiki/Discrete_derivative

.

Cũng theo tính chất của đạo hàm:

$F'(x) = \sum_{i=1..n} f_i'(x)$

Ví dụ: hàm màu tím là đạo hàm của hàm màu xanh



Ta tính đạo hàm thêm lần nữa:

$f_i''(x) = 2 * C(i)$ nếu $x = H(i+1)$, ngược lại $f_i''(x) = 0$.

Cũng theo tính chất của đạo hàm:

$F''(x) = \sum_{i=1..n} f_i''(x)$

Hàm $f_i''(x)$ chính là mấu chốt để giải bài toán này: hàm $f_i''(x)$ chỉ khác 0 tại đúng 1 điểm, do đó ta dễ dàng "tính" được $f_i''(x)$ trong $O(1)$.

$\Rightarrow$ ta có thể tính được $F''(x)$ trong $O(N)$ rất đơn giản như sau:

```
vector<int> F2(MAX_H + 1); // F2[x] = F''(x)
for (int i = 0; i < N; ++i) {
  // f''i(x) = 2*C(i) at x = H(i)+1
  F2[H[i] + 1] += 2 * C[i];
}
```

Dựa vào định nghĩa của đạo hàm ($f'(x) = f(x) - f(x-1)$), từ $F''(x)$ ta có thể dễ dàng tính được $F'(x)$ trong $O(max_H)$:

```
vector<int> F1(MAX_H + 1); // F1[x] = F'(x)
// Tính F'(0)
for (int i = 0; i < N; ++i) { F1[0] -= C[i]; }
// Tính F'(x)
for (int x = 1; x <= MAX_H; ++x) {
  F1[x] = F1[x - 1] + F2[x];
}
```

Tương tự, ta có thể tính được $F(x)$ trong $O(max_H)$:

```
vector<int> F(MAX_H + 1);
// Tính F(0)
for (int i = 0; i < N; ++i) {
  F[0] += H[i] * C[i];
}
// Tính F(x)
for (int x = 1; x <= MAX_H; ++x) {
  F[x] = F[x - 1] + F1[x];
}
```

Đến đây ta đã thu được thuật toán với độ phức tạp $O(N + max_H)$!

[91] https://en.wikipedia.org/wiki/Convex_function

Bonus

Sử dụng đạo hàm, ta cũng có thể chứng minh hàm $F(x)$ là hàm lồi, và do đó cách chặt tam phân trình bày ở trên là chính xác.

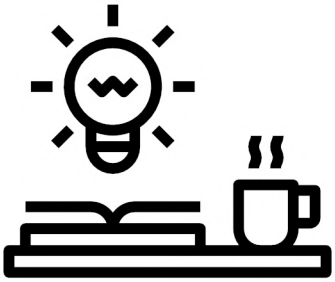
Một hàm số là hàm lồi trong đoạn $[l, r]$ nếu đạo hàm bậc 2 của nó luôn lớn hơn hoặc bằng 0 trong khoảng $[l, r]$. (Một cách hiểu trực quan: có thể thấy nếu đạo hàm bậc 2 không âm $\Rightarrow$ đạo hàm bậc 1 là hàm tăng dần $\Rightarrow$ hàm số luôn "cong cong" về phía trên). (Xem thêm trên trang Wiki về hàm lồi [91])

Ở bài này, dễ thấy $f_i(x)$ luôn lớn hơn hoặc bằng 0, do đó $F''(x) = \sum f_i(x)$ cũng luôn lớn hơn hoặc bằng 0.

Hình vẽ chụp từ Geogebra:
https://www.geogebra.org/calculator/pcyur4yk

Cách làm "lạc rang"

Trần Xuân Bách
12A1 Tin, Trường THPT Chuyên Khoa học Tự nhiên



Giới thiệu

Chắc hẳn khi học toán các bạn đã phải vẽ đồ thị của một đa thức P có bậc 1, 2 rồi. Cách làm phổ biến nhất là ta vẽ ra một vài điểm $(x, P(x))$ trên giấy, rồi nối chúng lại với nhau ~~và hi vọng nhìn nó đúng~~. Vậy đã bao giờ bạn nghĩ đến bài toán ngược lại:

> Từ một số điểm $(x_i, P(x_i))$ cho trước, những đa thức $P(x)$ nào có đồ thị đi qua các điểm đó?

Có thể chứng minh được rằng có vô hạn đa thức P thỏa mãn. Tuy nhiên, việc tìm đa thức P có bậc nhỏ nhất (**nội suy - interpolate**) thì khó hơn nhiều. Vào năm 1795, nhà toán học Joseph-Louis Lagrange đưa ra một công thức tổng quát để nội suy ra đa thức này, gọi là **nội suy Lagrange**. Phương pháp này được dùng nhiều trong lập trình thi đấu, và cũng được dùng trong mật mã học, với ví dụ điển hình là thuật toán Shamir [92] .

Nội suy Lagrange

Ở cốt lõi của nội suy là định lí sau:

Định lí 1

> Với mọi số tự nhiên $n \geq 0$ và mọi tập $n+1$ điểm $(x_1, y_1), (x_2, y_2), \ldots, (x_{n+1}, y_{n+1})$ với $x_1 < x_2 < \ldots < x_{n+1}$, **tồn tại duy nhất** một đa thức $P(x)$ bậc n sao cho $P(x_i) = y_i$ với mọi i.

Ta sẽ chứng minh hai phần của định lí này: sự **tồn tại** của đa thức p và tính **duy nhất** của nó.

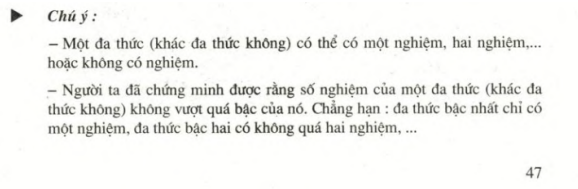
Chứng minh tính duy nhất

Ta giả sử phản chứng: Tồn tại hai đa thức P và Q khác nhau đều thỏa mãn điều kiện trên.

Xét đa thức $R = P - Q$, ta có:

- Vì P và Q đều có bậc n, nên R có bậc tối đa là n.
- Với mọi i, $R(x_i) = P(x_i) - Q(x_i) = y_i - y_i = 0$, vậy R có $n+1$ nghiệm $x_0, x_1, \ldots, x_n$.

Theo Định lí cơ bản của đại số [93] , R phải là đa thức không, vậy $P = Q$ (vô lí với giả sử phản chứng).



Chú ý :

– Một đa thức (khác đa thức không) có thể có một nghiệm, hai nghiệm,... hoặc không có nghiệm.

– Người ta đã chứng minh được rằng số nghiệm của một đa thức (khác đa thức không) không vượt quá bậc của nó. Chẳng hạn : đa thức bậc nhất chỉ có một nghiệm, đa thức bậc hai có không quá hai nghiệm, ...

47

Trích SGK toán lớp 7 tập 2

Vậy tồn tại tối đa một đa thức P thỏa mãn điều kiện của đề bài.

Chứng minh sự tồn tại

Ta sẽ chỉ ra một cách để xây dựng đa thức P thỏa mãn. Ý tưởng ở đây là ta sẽ tách P thành n đa thức con:

$$P = P_1 + P_2 + ... + P_{n+1}$$

Trong đó, mỗi đa thức con P_i sẽ thỏa mãn:

- $P_i(x_i) = y_i$
- $P_i(x_j) = 0 \ (\forall j \neq i)$.

Ta sẽ thêm các thừa số $(x - x_j)$ vào P_i để khiến $P_i(x_j) = 0$, rồi sau đó nhân nó với một giá trị thích hợp để $P_i(x_i) = y_i$.

Cuối cùng công thức tổng quát cho P_i và P là:

$$\begin{aligned}
P_i(x) = \ & y_i \times \frac{x - x_1}{x_i - x_1} \times \cdots \times \frac{x - x_{i-1}}{x_i - x_{i-1}} \times \\
& \times \frac{x - x_{i+1}}{x_i - x_{i+1}} \times \cdots \times \frac{x - x_{n+1}}{x_i - x_{n+1}} \\
= \ & y_i \prod_{j \neq i} \frac{x - x_j}{x_i - x_j} \\
\Rightarrow P(x) = \ & y_1 \prod_{j \neq 1} \frac{x - x_j}{x_1 - x_j} + \cdots + \\
& + y_{n+1} \prod_{j \neq n+1} \frac{x - x_j}{x_{n+1} - x_j} \\
= \ & \sum_{i=1}^{n+1} y_i \prod_{j \neq i} \frac{x - x_j}{x_i - x_j}
\end{aligned}$$

Bạn đọc có thể tự kiểm chứng rằng đa thức P thỏa mãn điều kiện của định lí trên. Cách xây dựng $P(x)$ trên được gọi là nội suy Lagrange.

[92] https://en.wikipedia.org/wiki/Shamir%27s_Secret_Sharing

[93] https://vi.wikipedia.org/wiki/Định_lý_cơ_bản_của_đại_số

Cài đặt mẫu

C++

```cpp
const int N = 1000 + 5;

struct Point {
  double x, y;
} data[N];

// Tính giá trị của P tại x
double interpolation(int n, double x) {
  double ans = 0;
  for (int i = 1; i <= n + 1; i++) {
    // Tính giá trị của P_i tại x
    double val = data[i].y;
    for (int j = 1; j <= n + 1; j++) {
      if (i == j) continue;
            val *= (x - data[j].x) / (data[i].x -
↪  data[j].x;
    }
    ans += val;
  }
  return ans;
}
```

Python

```python
def interpolation(data, x):
    ans = 0
    for i in range(len(data)):
        val = data[i].y
        for j in range(len(data)):
            if i == j:
                continue
            val *= (x - data[j].x) / (data[i].x -
            ↪  data[j].x)
        ans += val
    return ans
```

Phân tích

Cách cài đặt trên có độ phức tạp là $\mathcal{O}(n^2)$ (2 vòng `for` lồng nhau).

Thuật toán SSS

Thuật toán Chia sẻ bí mật Shamir (Shamir's Secret Sharing - SSS) là một trong những ứng dụng điển hình nhất của phép nội suy đa thức. SSS là một thuật toán dùng để chia sẻ một *bí mật* nào đó cho nhiều người, sao cho không người nào nắm giữ được bất kì thông tin quan trọng nào của bí mật đó. Để làm được điều này, SSS tách bí mật ra thành n phần, mà chỉ khi ta nắm giữ được ít nhất k phần thì ta mới xây dựng lại được bí mật này. Thuật toán này có *tính bảo mật tuyệt đối*, tức là một kẻ gian không thể nào tìm được bí mật đó kể cả khi hắn có vô hạn thời gian và năng lực tính toán.

Phương thức hoạt động

Giả sử ta đang muốn chia sẻ một bí mật S ra thành n phần $S_1, S_2, \ldots, S_n$, sao cho chỉ khi ta nắm giữ được k phần thì ta mới tìm được S, và nếu ta chỉ nắm giữ được $k-1$ trở xuống thì ta sẽ không bao giờ tìm được **bất kì thông tin gì** từ S. Để dễ hiểu cho bạn đọc ta sẽ giả sử ta đang làm việc với các số nguyên.

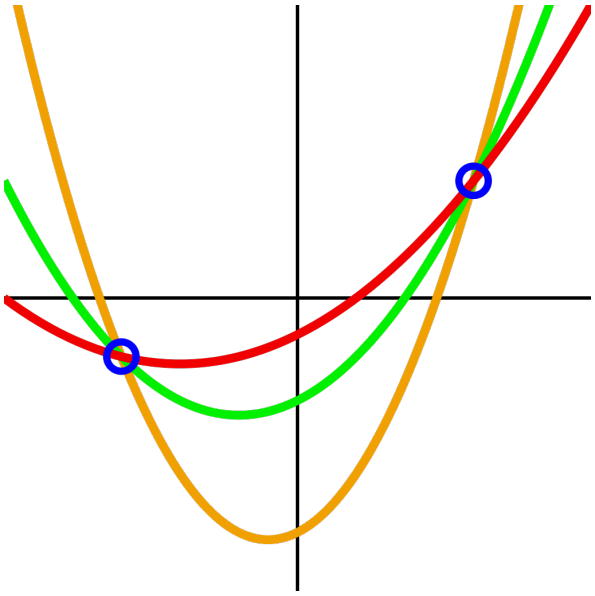
Ta sẽ xây dựng một đa thức $P(x) = p_0 + p_1 \times x^1 + \cdots + p_{k-1} \times x^{k-1}$ có bậc $k-1$, sao cho:

- $p_0 = S$.
- $p_1, \ldots, p_{k-1}$ là các số nguyên được chọn ngẫu nhiên.

Sau đó, mỗi phần S_i của bí mật sẽ là một cặp số $(i, P(i))$.

Nếu ta có k phần bí mật (x_1, y_1), (x_2, y_2), $\ldots$, (x_k, y_k), ta có thể dùng nội suy đa thức để tính $P(0) = \sum_{i=1}^{k} y_i \prod_{j \neq i} \frac{0 - x_j}{x_i - x_j}$. Nhận thấy $P(0)$ cũng bằng $p_0 + p_1 \times 0^1 + \cdots + p_{k-1} \times 0^{k-1} = p_0 = S$, vậy ta đã dựng lại được bí mật S.

Nếu ta chỉ sở hữu $k-1$ phần bí mật, với mỗi giá trị y_k bất kì, ta có thể sinh ra được một đa thức P thoả mãn. Do vậy, có vô vàn đa thức P thoả mãn, với vô vàn các giá trị khác nhau cho $P(0)$. Điều này đồng nghĩa với việc nắm giữ $\leq k-1$ phần không đưa ra bất kì thông tin gì về bí mật.



Có vô số đa thức bậc hai đi qua hai điểm cho trước như trên

Trên thực tế, các giá trị S, S_i, p_i, x_i, y_i thường được tính theo modulo của một số nguyên tố đủ lớn p, với lí do sẽ được đề cập sau.

Ví dụ: CTF BabySSS

Giới thiệu

Capture the flag (CTF) [94] là một dạng kì thi khá giống với lập trình thi đấu, với mục đích là thử thách các kĩ năng nhận diện và xử lí lỗ hổng bảo mật của thí sinh. Để giải được bài, thí sinh cần phải tìm được một "lá cờ" (flag) được giấu trong một trang web hoặc phần mềm, bằng cách lợi dụng các lỗ hổng bảo mật được cố tình đưa vào từ ban ra đề.

CTF không chỉ đơn giản và thân thiện hơn với thí sinh mới so với lập trình thi đấu, mà còn phổ cập cho thí sinh các kiến thức cơ bản về bảo mật thông tin nói riêng và máy tính nói chung.

Mình và một vài người bạn có tham gia một kì thi vào ngày 25-27/11, mang tên HITCON CTF [95] . Bài thi dễ nhất của phần mật mã mang tên BabySSS, và như bạn đọc có thể đoán, ta phải tìm ra một lỗ hổng nào đó trong phần mềm cài đặt thuật toán SSS của ban tổ chức để từ đó lấy được flag.

Đề bài

Link [96]

> I implemented a toy Shamir's Secret Sharing [97] for fun. Can you help me check is there any issues with this?

Ta cũng được cho thêm một file nén `babysss.tar.gz`, trong đó bao gồm hai file:

- `chall.py`

```
from random import SystemRandom
from Crypto.Cipher import AES
from hashlib import sha256
from secret import flag

rand = SystemRandom()


def polyeval(poly, x):
    return sum([a * x**i for i, a in enumerate(poly)])


DEGREE = 128
SHARES_FOR_YOU = 8  # I am really stingy :)

poly = [rand.getrandbits(64) for _ in range(DEGREE + 1)]
shares = []
for _ in range(SHARES_FOR_YOU):
    x = rand.getrandbits(16)
    y = polyeval(poly, x)
    shares.append((x, y))
print(shares)

secret = polyeval(poly, 0x48763)
key = sha256(str(secret).encode()).digest()[:16]
cipher = AES.new(key, AES.MODE_CTR)
print(cipher.encrypt(flag))
print(cipher.nonce)
```

- `output.txt`[98].

```
[(41458, 30158948896...
```

Hướng suy nghĩ

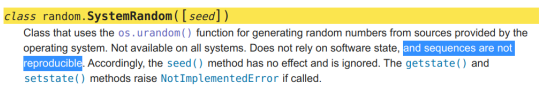
Phân tích code

Đầu tiên, hãy đọc từng đoạn code một:

- Dòng 1-6

```
from random import SystemRandom
from Crypto.Cipher import AES
from hashlib import sha256
from secret import flag

rand = SystemRandom()
```

Đây là những thư viện mà `chall.py` sẽ sử dụng - có thể thấy ta import AES và sha256 là cách mã hóa và hash phổ biến. Ta cũng thấy có một biến `flag` được import từ module `secret`, ta đoán rằng đây chính là flag mà mình cần tìm. Bình thường CTF cũng có vài bài lợi dụng việc tìm seed được dùng cho random để mô phỏng lại việc mã hóa `flag`, tuy nhiên khi mình tra tài liệu thì nó bảo là `SystemRandom` không mô phỏng lại được luôn.


class random.SystemRandom([seed])

Class that uses the os.urandom() function for generating random numbers from sources provided by the operating system. Not available on all systems. Does not rely on software state, and sequences are not reproducible. Accordingly, the seed() method has no effect and is ignored. The getstate() and setstate() methods raise NotImplementedError if called.

Tài liệu chính thức của Python về `SystemRandom`

- Dòng 9-22

```
def polyeval(poly, x):
    return sum([a * x**i for i, a in
    ↪  enumerate(poly)])


DEGREE = 128
SHARES_FOR_YOU = 8  # I am really stingy :)

poly = [rand.getrandbits(64) for _ in range(DEGREE +
↪  1)]
shares = []
for _ in range(SHARES_FOR_YOU):
    x = rand.getrandbits(16)
    y = polyeval(poly, x)
    shares.append((x, y))
print(shares)
```

  - Hàm `polyeval` là để tính giá trị của đa thức `poly` tại điểm `x`.

[94] https://en.wikipedia.org/wiki/Capture_the_flag_(cybersecurity)
[95] https://ctf2022.hitcon.org/
[96] https://ctf2022.hitcon.org/dashboard/#8
[97] https://en.wikipedia.org/wiki/Shamir%27s_Secret_Sharing
[98] Đây là file có nội dung rất dài, các bạn vui lòng tải file từ trang web để xem đầy đủ

- Hai biến sau cho biết rằng đa thức của chúng ta có bậc là 128, nhưng ta chỉ được biết đúng 8 điểm. Về mặt lí thuyết ta phải biết đúng 128 điểm thì ta mới tìm được đa thức `poly`.
- Các hệ số của `poly` là các số 64-bit ngẫu nhiên, tức là các số nằm trong khoảng $[0, 2^{64})$.
- Các giá trị x của cặp điểm $(x, y = \text{poly}(x))$ là số 16-bit ngẫu nhiên.
- Cuối cùng, ta được cho biết thông tin của 8 điểm ngẫu nhiên ở file `output.txt`

- Dòng 24-28

```
secret = polyeval(poly, 0x48763)
key = sha256(str(secret).encode()).digest()[:16]
cipher = AES.new(key, AES.MODE_CTR)
print(cipher.encrypt(flag))
print(cipher.nonce)
```

Gọi `secret` là giá trị của `poly` tại $x = 0\text{x}48673$. Ta mã hóa `flag` cần tìm bằng AES với `key` ở đây là hash SHA256 của `secret`, rồi in ra `flag` và phần `nonce` được thêm vào.
Do phần này không quan trọng đến lời giải nên mình sẽ tạm bỏ qua cách xử lí phần này. Bạn đọc chỉ cần biết rằng nếu mình có giá trị của `secret`, thì kết hợp với hai giá trị được in ra ở phía trên, có thể tìm lại được `flag` ban đầu.

Tìm lỗ hổng

Khi mình đọc trang Wikipedia của thuật toán SSS đã được link trong đề bài, mình để ý đến một đoạn nói về "Vấn đề về việc tính toán bằng số nguyên". Cụ thể hơn, câu đầu của đoạn như sau:

> Phiên bản đơn giản hơn của thuật toán trên, mà dùng số nguyên để tính toán thay vì dùng modulo, vẫn hoạt động. Tuy nhiên, nó lại có một vấn đề bảo mật: Kẻ xấu sẽ có thêm thông tin về S với mỗi phần mà hắn có được.

Sau đó, có một ví dụ nói về việc khi chỉ có được 2 điểm trong khi $k = 3$, ta đã có thể rút gọn số giá trị có thể của S về 150 số bằng phương pháp đồng dư. Đoạn code trên cũng tính toán bằng số nguyên, vậy ta có thể làm tương tự được không?

Hãy quay lại các cặp giá trị $(x_1, y_1), \ldots, (x_8, y_8)$, ta có:

$$\begin{aligned} y_i &= \text{poly}(x_i) \\ &= \text{poly}[0] + \text{poly}[1] \times x_i + \cdots + \\ &\quad + \text{poly}[128] \times x_i^{128} \\ \Rightarrow y_i &\equiv \text{poly}[0] \pmod{x_i} \end{aligned}$$

Vậy ta có 8 biểu thức đồng dư như sau:

$$\begin{aligned} \text{poly}[0] &\equiv y_1 \pmod{x_1} \\ \text{poly}[0] &\equiv y_2 \pmod{x_2} \\ &\ldots \\ \text{poly}[0] &\equiv y_8 \pmod{x_8} \end{aligned}$$

Chắc hẳn nhiều bạn sẽ nhận ra đây chính là định lí thặng dư Trung Hoa [99] . Sử dụng định lí này, ta sẽ tìm được giá trị của $\text{poly}[0] \bmod \text{lcm}(x_1, x_2, \ldots, x_8)$. Do $\text{poly}[0]$ là một số ngẫu nhiên có 64 bit, nên nếu giá trị của hàm lcm trên không nhỏ hơn 2^{64} thì ta đã tìm được giá trị thỏa mãn duy nhất của $\text{poly}[0]$.

Để kiểm tra điều kiện trên, ta có thể viết một đoạn code nhỏ `helper.py` như sau:

```
import math

shares = [(41458, ...

shares_x = [point[0] for point in shares]
print("shares_x =", shares_x)
print("lcm =", math.lcm(*shares_x))
print("2^64 =", 2 ** 64)
```

```
bdc07$ python3 helper.py
shares_x = [41458, 3389, 20016, 50683, 6445, 1359, 45286, 5649]
lcm = 29571522287147838817799689634 40
2^64 = 18446744073709551616
```

Kết quả của đoạn code trên

Vậy ta đã tìm được giá trị của $\text{poly}[0]$, làm thế nào để tìm nốt các hệ số còn lại của đa thức? Ta sẽ dùng một mẹo như sau:

$$\begin{aligned} & y_i && = \text{poly}[0] + \text{poly}[1] \times x_i + \\ & && \quad + \cdots + \text{poly}[128] \times x_i^{128} \\ \iff & y_i - \text{poly}[0] && = \text{poly}[1] \times x_i + \\ & && \quad + \cdots + \text{poly}[128] \times x_i^{128} \\ \iff & \frac{y_i - \text{poly}[0]}{x_i} && = \text{poly}[1] + \text{poly}[2] \times x_i + \\ & && \quad + \cdots + \text{poly}[128] \times x_i^{127} \end{aligned}$$

Ta có thể thấy vế phải biến thành một đa thức có bậc là 127 với các hệ số lần lượt là $\text{poly}[1], \ldots, \text{poly}[128]$. Làm tương tự như $\text{poly}[0]$, ta sẽ lần lượt tìm được các giá trị $\text{poly}[1], \ldots, \text{poly}[128]$.

[99] https://en.wikipedia.org/wiki/Chinese_remainder_theorem

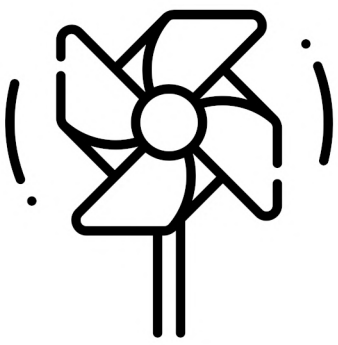
Lời giải

```
import ast
from hashlib import sha256
from Crypto.Cipher import AES

# Đọc giá trị từ file output.txt
with open("output.txt") as f:
    shares = ast.literal_eval(f.readline())
    ciphertext = ast.literal_eval(f.readline())
    nonce = ast.literal_eval(f.readline())

# Mình đổi loại của shares[i] từ tuple sang list
# để có thể sửa giá trị của shares về sau
for i in range(len(shares)):
    shares[i] = list(shares[i])

def polyeval(poly, x):
    return sum([a * x**i for i, a in enumerate(poly)])

DEGREE = 128
SHARES_FOR_YOU = 8  # I am really stingy :)

poly = []
for d in range(DEGREE + 1):
    values = []
    mods = []
    for [x, y] in shares:
        values.append(y % x)
        mods.append(x)

    # Dùng hàm CRT của Sagemath để tính giá trị của
    ↪  poly[0]
    val = CRT(values, mods)
    poly.append(val)

    # Mẹo đã nói ở trên
    for i in range(len(shares)):
        shares[i][1] = (shares[i][1] - val) //
        ↪  shares[i][0]

# Tìm lại flag sau khi biết được secret
secret = polyeval(poly, int(0x48763))
key = sha256(str(secret).encode()).digest()[:16]
cipher = AES.new(key, AES.MODE_CTR, nonce = nonce)
flag = cipher.decrypt(ciphertext)
print(flag)
```

Bổ đề "cháy cạnh"

(Bổ đề Burnside & Định lý liệt kê Pólya)

Nguyễn Hoàng Dũng
11 Tin, Trường THPT Chuyên Hà Nội - Amsterdam



Giới thiệu

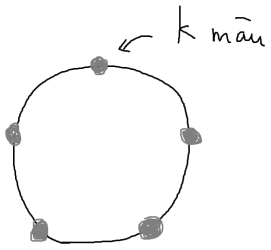
Nếu bạn đã hoặc đang là một học sinh phổ thông, có lẽ câu hỏi dưới đây sẽ không quá xa lạ gì với các bạn:

> Đếm số đồng phân của các chất sau: C_6H_4ClBr, $C_6H_2Cl_2Br_2$, $C_6H_2ICl Br_2$.

...

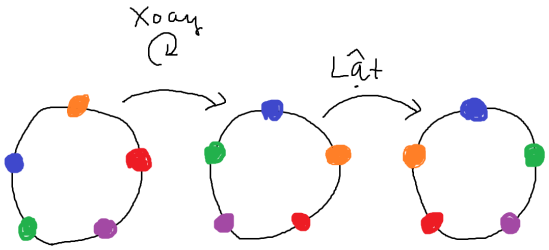
Mình sẽ lấy một ví dụ khác nhẹ nhàng hơn:

> Đếm số chuỗi hạt gồm 5 hạt được tạo bởi các hạt màu đỏ, tím, vàng. Hay tổng quát hơn là đếm số chuỗi hạt gồm n hạt được tạo bởi k màu.



Do mỗi hạt có k cách tô nên hiển nhiên n hạt sẽ có k^n cách tô.

Thực tế thì kết quả không đơn giản như vậy, chuỗi hạt của chúng ta có thể được xoay hoặc lật nên cách đếm ngây thơ sẽ không đúng.



Vậy, chúng ta sẽ giải quyết những bài toán đếm này như thế nào?

Lý thuyết nhóm, hay chính xác hơn, **bổ đề Burnside** sẽ cho chúng ta một phương pháp phù hợp để giải những dạng bài như trên.

Bổ đề Burnside

Công thức của Bổ đề Burnside được chứng minh bởi Burnside vào năm 1897, nhưng trước đó nó đã được khám phá ra vào năm 1887 bởi Frobenius, và sớm hơn nữa vào năm 1845 bởi Cauchy. Do đó thi thoảng nó cũng được gọi là Bổ đề Cauchy-Frobenius. Thú vị hơn là chính bản thân Burnside cũng viết trong cuốn sách của mình rằng khám phá này là của Frobenius.

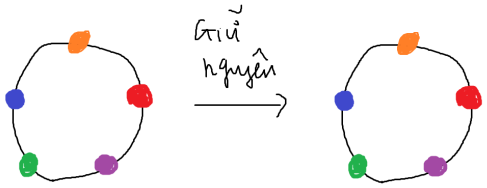
Kiến thức cần biết (Giới thiệu qua về lý thuyết nhóm)

Do bổ đề Burnside là một kết quả từ lý thuyết nhóm, nên ta cần hiểu một số định nghĩa sau đây:

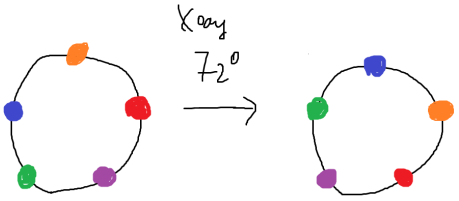
Nhóm, nhóm đối xứng

Trước hết, ta cần hiểu rõ về các phép biến đổi đối xứng, nghĩa là các phép biến đổi mà dưới tác động của chúng, hai chuỗi hạt được coi là một. Các phép biến đổi này sẽ tạo thành một **nhóm**.

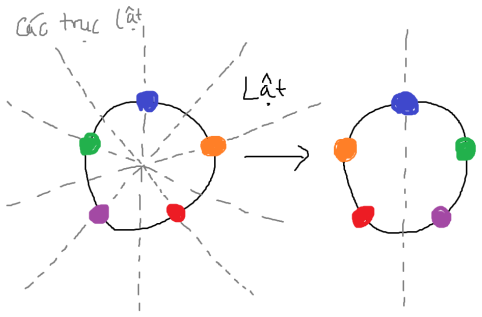
Phép biến đổi đầu tiên, quan trọng nhất nhưng cũng kém thú vị nhất là phép biến đổi đơn vị đơn giản là không làm gì cả



Phép biến đổi thứ hai là phép xoay:



Và phép biến đổi cuối cùng là phép lật:



Khi nói về nhóm, ta cũng phải quan tâm đến phép toán của nhóm, nhưng trong trường hợp này, kết quả của sự kết hợp (phép hợp) các phép xoay và lật với nhau, cũng ra các phép xoay và lật.

Định nghĩa chính xác của nhóm là:

> Một **nhóm** là một cặp $(G, +)$, với G là một tập không rỗng và $+ : GGßG$ là một phép toán thỏa mãn điều kiện:
>
> - **Tính kết hợp**: $(a + b) + c = a + (b + c)$ với mọi $a, b, c \in G$.
> - **Phần tử đơn vị**: Tồn tại $0 \in G$ sao cho $0 + a = a + 0 = a$ với mọi $a \in G$.
> - **Phần tử nghịch đảo**: Với mọi $a \in G$, tồn tại một phần tử nghịch đảo $-a \in G$ sao cho $a + (-a) = (-a) + a = 0$.

Nghĩ theo góc độ các phép biến đổi đối xứng, phần tử đơn vị có nghĩa là tồn tại phép biến đổi không làm gì cả, phần tử nghịch đảo nghĩa là mỗi phép biến đổi có cách làm ngược lại, và tính kết hợp nghĩa là các phép biến đổi hoạt động "ổn" khi ta cho thêm dấu ngoặc vào.

Trong trường hợp của chúng ta thì, nhóm G gồm các phép biến đổi đối xứng, phép toán là hợp của hai phép biến đổi và phần tử đơn vị là không làm gì. Trong lý thuyết nhóm, đây còn được gọi là nhóm nhị diện D_n.

Ngoài ra còn nhiều ví dụ về nhóm khác như, tập số nguyên với phép cộng $(\mathbb{Z}, +)$, tập số thực khác 0 với phép nhân $(\mathbb{R} \setminus \{0\}, \times)$, ...

Một **nhóm đối xứng** là một nhóm mà phần tử của nó là các phép biến đổi của một đối tượng, và phép toán là phép toán hợp $\circ$ (giống như hợp của các hàm), ví dụ như phép xoay / lật như trên.

Một nhóm đối xứng đáng ra được viết là $(G, \circ)$ với tập các phép biến đổi và phép hợp được viết rõ ràng, nhưng phép hợp thường được ngụ ý hiểu, do đó một nhóm đối xứng thường được viết đơn giản là G.

Tác động nhóm

Ta có một tập S, ta nói nhóm G *tác động* lên tập S nếu với mọi phần tử g trong G và phần tử s trong S, tồn tại $g \cdot s$ trong S từ tác động của G lên s, sao cho thỏa mãn 2 điều kiện sau:

1. Phần tử đơn vị e là phần tử đơn vị trên S. Nghĩa là, với mọi s trong S, $e \cdot s = s$.
2. Tác động nhóm phù hợp với phép toán của nhóm. Nghĩa là, với mọi g, h trong G và s trong S, $(gh) \cdot s = g \cdot (h \cdot s)$

Xét với trường hợp của chúng ta, ta sẽ xét tập S của tất cả các cách tô màu một chuỗi hạt (cả k^n theo cách đếm ngây thơ).

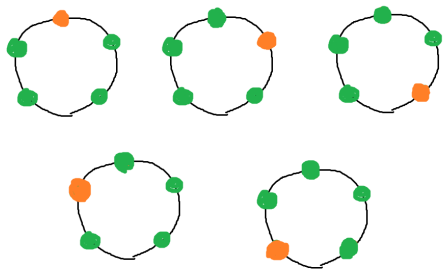
Và bây giờ nhóm D_n bên trên tác động lên S. Một phép xoay một cách tô màu sẽ ra một phép tô màu, và một phép lật cũng thế.

Với 2 cách tô màu trong S, nếu có thể biến từ một cách sang cách khác bằng phép xoay và lật, thì nó sẽ ứng với cùng một chuỗi hạt. Nghĩa là tác động của D_n lên S có thể biến đổi một cái thành cái kia. Trong lý thuyết nhóm, ta nói rằng hai cách tô màu ở trong cùng một **quỹ đạo** của G.

Nói một cách chính xác:

> Quỹ đạo của một phần tử s là tất cả các phần tử s' sao cho s' có thể biến đổi được từ S bằng một tác động, nghĩa là tồn tại g trong G sao cho $g \cdot s = s'$

Vậy thì hai cách tô màu ở trong cùng một quỹ đạo (có thể biến đổi từ một cái sang cái còn lại chỉ bằng xoay và lật) sẽ ứng với cùng một chuỗi hạt. Một quỹ đạo sẽ ứng với duy nhất một chuỗi hạt, và dễ thấy với mỗi chuỗi hạt có duy nhất một quỹ đạo ứng với nó. Vậy, để đếm số chuỗi hạt, ta sẽ đếm số quỹ đạo.



Vậy chúng ta đếm số lượng quỹ đạo như thế nào?

Phát biểu bổ đề

Bổ đề Burnside:

Gọi G là nhóm hữu hạn các phép biến đổi tác động lên tập X. Gọi X/G là tập các quỹ đạo của X (mỗi phần tử của X/G là một quỹ đạo của X). Với mọi phần tử $g \in G$, gọi X^g là tập các điểm bất biến (cố định) của X đối với phép biến đổi g ($X^g = \{x \in X : g \cdot x = x\}$). Khi đó,

$$|X/G| = \frac{1}{|G|} \sum_{g \in G} |X^g|$$

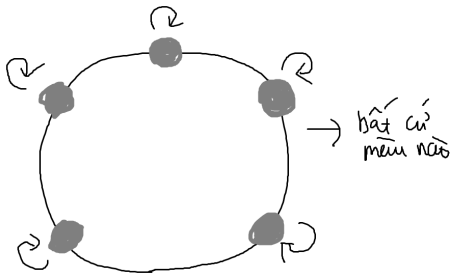
Chứng minh bổ đề

Việc chứng minh bổ đề Burnside không quá quan trọng đối với các ứng dụng thực tế, vì vậy nó sẽ được coi là bài tập cho bạn đọc.

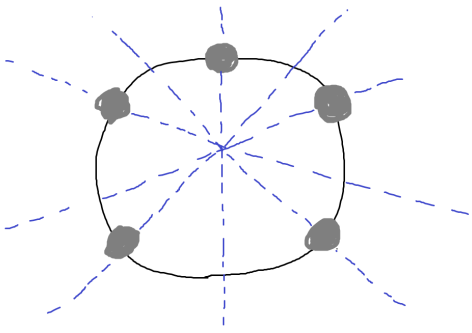
Ứng dụng

Với bổ đề Burnside, chúng ta có thể giải quyết bài toán. Để cho đơn giản, ta hãy lấy $n = 5$. Khi đó, với mọi phần tử của D_5, ta phải đếm số lượng điểm mà nó cố định (số cách tô tương đương sau khi được xoay hoặc lật).

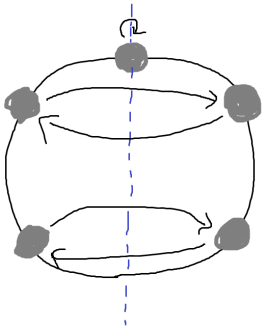
Đầu tiên, xét phần tử đơn vị, vì nó không làm gì nên nó sẽ cố định tất cả k^5 cách tô màu:



Sau đó xét tới phép lật. Mỗi phép lật có trục đi qua các hạt như sau:

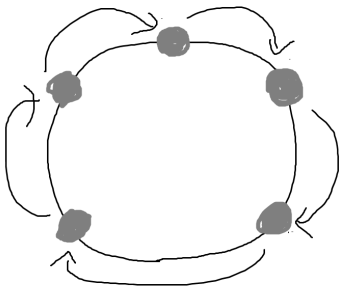


Và bốn hạt còn lại được chia thành hai nhóm, đối xứng với nhau qua trục:



Vậy để một cách tô màu được cố định bởi phép lật, hạt ở trên trục có thể tô bất cứ màu nào, và các hạt khác phải cùng màu với hạt đối xứng với nó. Do đó với mỗi trục ta cố định được k^3 phần tử. Vì có 5 trục đi qua mỗi hạt, ta có tổng cộng $5k^3$.

Cuối cùng, xét tới phép xoay. Đầu tiên là xoay 72 độ theo chiều kim đồng hồ (chuyển tất cả các hạt sang hạt bên cạnh theo chiều kim đồng hồ):



Để cách tô được cố định bởi phép xoay này, hạt 1 phải giống hạt 2, hạt 2 phải giống hạt 3, 3 giống 4, 4 giống 5 và 5 giống 1. Do đó tất cả các hạt phải có cùng màu, nghĩa là có n cách tô.

Với ba cách xoay còn lại chúng ta có thể lập luận tương tự, vậy tổng lại phép quay có $4k$ cách tô.

Theo bổ đề Burnside, số lượng quỹ đạo, cũng là số lượng chuỗi hạt, là:

$$\frac{k^5 + 5k^3 + 4k}{10}$$

Định lý liệt kê Pólya

Ta có thể phần nào thấy rằng việc chọn $k = 5$ là một số nguyên tố khiến cho phép xoay chỉ có thể tô cùng một màu, vậy nếu k khác đi thì sao, hoặc với k lớn hơn và ta không còn có thể xét tay như trên?

Định lý liệt kê Pólya sẽ giúp ta làm điều đó.

Định lý liệt kê Pólya là một tổng quát của bổ đề Burnside, và nó cũng cung cấp một công cụ thuận tiện hơn để tìm số lớp tương đương. Ở đây chúng ta chỉ nhắc đến một trường hợp đặc biệt của định lý liệt kê Pólya, trường hợp này sẽ rất hữu ích trong thực tế. Công thức chung của định lý sẽ không được nhắc đến.

Định lý này đã được Redfield phát hiện trước Pólya vào năm 1927, nhưng công bố của ông không được các nhà toán học chú ý. Pólya độc lập đạt được kết quả tương tự vào năm 1937, và công bố của ông thành công hơn.

Với trường hợp của chúng ta, ta đánh số các hạt từ 1 đến k và coi như các phép xoay và lật là các **hoán vị bất biến** (nghĩa là hoán vị không thay đổi đối tượng mà chỉ là cách biểu diễn), phép toán là phép hợp hai hoán vị. Và nhóm G khi đó ta sẽ gọi là **nhóm các hoán vị bất biến**.

Tìm tất cả các hoán vị bất biến như vậy với định nghĩa của một đối tượng là một bước quan trọng để áp dụng bổ đề Burnside và định lý liệt kê Pólya. Những hoán vị bất biến này phụ thuộc vào từng bài toán cụ thể nhưng trong hầu hết các trường hợp, việc tìm một số hoán vị "cơ bản" bằng tay là đủ để sinh ra tất cả các hoán vị khác bằng máy.

Sau đó, ta có thể phát biểu định lý như sau:

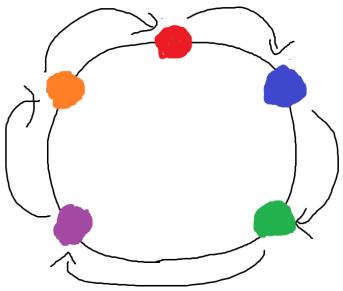
Phát biểu định lý

> **Trường hợp đặc biệt của định lý liệt kê Pólya**:
>
> Ta gọi $C(\pi)$ là số chu trình trong hoán vị π, k là số giá trị mà mỗi phần tử có thể nhận. Khi đó,
>
> $$|X/G| = \frac{1}{|G|} \sum_{\pi \in G} k^{C(\pi)}$$

Chứng minh định lý

Định lý liệt kê Pólya là hệ quả trực tiếp của bổ đề Burnside, vì vậy việc chứng minh nó sẽ được coi là bài tập cho bạn đọc.

Ứng dụng: Tô màu vòng cổ



> Đếm số vòng cổ khác nhau từ n hạt, mỗi hạt có thể được tô bằng một trong các màu k. Khi so sánh hai vòng cổ, chúng có thể được xoay, nhưng không được đảo ngược (tức là chỉ cho phép dịch chuyển vòng tròn).

(Đây là phiên bản dễ hơn của bài toán ở trên mà không có phép lật.)

Trong bài toán này ta có thể tìm ngay được nhóm các hoán vị bất biến:

$$\begin{aligned} \pi_0 &= 123\ldots n \\ \pi_1 &= 23\ldots n1 \\ \pi_2 &= 3\ldots n12 \\ &\ldots \\ \pi_{n-1} &= n123\ldots \end{aligned}$$

Ta hãy tìm một công thức rõ ràng để tính toán $C(\pi_i)$. Trước tiên, chúng tôi lưu ý rằng hoán vị π_i có ở vị trí j-th giá trị $i + j$ (tính theo mô-đun n). Nếu ta xét cấu trúc chu trình cho π_i, ta thấy rằng 1 chuyển thành $1 + i$, $1 + i$ chuyển thành $1 + 2i$, chuyển thành $1 + 3i$, v.v., cho đến một số có dạng $1 + kn$. Lập luận tương tự cho các phần tử còn lại. Do đó, chúng ta thấy rằng tất cả các chu trình đều có cùng độ dài, cụ thể là $\frac{\text{lcm}(i,n)}{i} = \frac{n}{\gcd(i,n)}$. Vì vậy, số chu kỳ trong π_i sẽ bằng $\gcd(i, n)$.

Thay các giá trị này vào định lý liệt kê Pólya, ta thu được công thức:

$$\frac{1}{n} \sum_{i=1}^{n} k^{\gcd(i,n)}$$

Bạn có thể để công thức này ở dạng này hoặc bạn có thể đơn giản hóa nó hơn nữa. Hãy thay đổi thứ tự tính tổng để nó lặp trên tất cả các ước của n. Trong công thức ban đầu sẽ có nhiều số hạng tương đương: nếu i không phải là ước của n, thì có thể tìm được ước sau khi tính $\gcd(i, n)$. Do đó, đối

với mỗi ước số $d \mid n$ số hạng của nó $k^{\gcd(d,n)} = k^d$ sẽ xuất hiện trong tổng nhiều lần, tức là câu trả lời cho bài toán có thể được viết lại thành

$$\frac{1}{n} \sum_{d \mid n} C_d k^d,$$

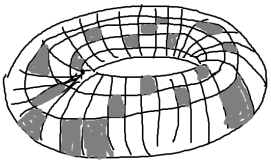
trong đó C_d là số các số i với $\gcd(i,n) = d$. Chúng ta có thể tìm được một biểu thức rõ ràng cho giá trị này. Bất kỳ số i nào như vậy đều có dạng $i = dj$ với $\gcd(j, n/d) = 1$ (nếu không thì $\gcd(i,n) > d$). Vì vậy, chúng tôi có thể đếm số j với tính chất này. Phi hàm Euler cho ta $C_d = \phi(n/d)$, và do đó ta có đáp án:

$$\frac{1}{n} \sum_{d \mid n} \phi\left(\frac{n}{d}\right) k^d$$

Cài đặt mẫu:

```
#include <bits/stdc++.h>
using namespace std;

const int MOD = (int)1e9 + 7;

int power(int a, int b) {
  int ans = 1;
  while (b > 0) {
    if (b % 2 == 1) { ans = 1LL * ans * a % MOD; }
    a = 1LL * a * a % MOD;
    b /= 2;
  }
  return ans;
}

int inverse(int a) { return power(a, MOD - 2); }

int main() {
  int t;
  cin >> t;

  for (int tc = 1; tc <= t; tc++) {
    int n, k;
    cin >> n >> k;

    int ans = 0;
    for (int i = 1; i <= n; i++) {
      ans = (ans + power(k, __gcd(i, n))) % MOD;
    }

    ans = 1LL * ans * inverse(n) % MOD;
    cout << "Case " << tc << ": " << ans << '\n';
  }
}
```

Ứng dụng: Tô màu hình xuyến



Thông thường, chúng ta không thể có được một công thức rõ ràng cho số lớp tương đương. Trong nhiều bài toán, số lượng hoán vị trong một nhóm có thể quá lớn để tính toán thủ công và không thể tính toán số chu trình trong chúng.

Trong trường hợp đó, ta nên tìm một số hoán vị "cơ bản" một cách thủ công để sinh ra toàn bộ nhóm G. Tiếp theo, chúng ta có thể viết một chương trình sẽ sinh tất cả các hoán vị của nhóm G, đếm số chu trình trong chúng và tính đáp án bằng công thức.

> Xét ví dụ về bài toán tô màu hình xuyến. Có một tờ giấy kẻ ca-rô $n \times m$ ($n < m$), một số ô được tô màu đen.
>
> Sau đó, một hình trụ thu được bằng cách dán hai mặt có chiều dài m lại với nhau. Sau đó, một hình xuyến thu được từ hình trụ bằng cách dán hai vòng tròn (trên và dưới) lại với nhau mà không xoắn. Nhiệm vụ là tính toán số lượng hình xuyến có màu khác nhau, giả sử rằng chúng ta không thể nhìn thấy các đường được dán và hình xuyến có thể được xoay.

Chúng ta lại bắt đầu với một tờ giấy $n \times m$. Dễ dàng thấy rằng các loại phép biến đổi sau bảo toàn lớp tương đương: sự dịch chuyển vòng tròn của các hàng, sự dịch chuyển vòng tròn của các cột và xoay tờ giấy $180°$. Cũng dễ dàng nhận thấy rằng các phép biến hình này có thể sinh ra toàn bộ nhóm các phép biến hình bất biến. Nếu bằng cách nào đó chúng ta đánh số các ô của tờ giấy, thì chúng ta có thể viết ba hoán vị p_1, p_2, p_3 tương ứng với các kiểu biến đổi này.

Tiếp theo, chỉ còn việc tạo ra tất cả các hoán vị thu được dưới dạng một tích. Hiển nhiên là tất cả các hoán vị như vậy đều có dạng $p_1^{i_1} p_2^{i_2} p_3^{i_3}$ trong đó $i_1 = 0 \dots m-1$, $i_2 = 0 \dots n-1$, $i_3 = 0 \dots 1$.

Cài đặt mẫu:

```
using Permutation = vector<int>;

void operator*=(Permutation &p,
                Permutation const &q) {
  Permutation copy = p;
  for (int i = 0; i < p.size(); i++)
    p[i] = copy[q[i]];
}

int countCycles(Permutation p) {
  int cnt = 0;
  for (int i = 0; i < p.size(); i++) {
    if (p[i] != -1) {
      cnt++;
      for (int j = i; p[j] != -1;) {
```

```
        int next = p[j];
        p[j] = -1;
        j = next;
      }
    }
  }
  return cnt;
}

int solve(int n, int m) {
  Permutation p(n * m), p1(n * m), p2(n * m),
    p3(n * m);
  for (int i = 0; i < n * m; i++) {
    p[i] = i;
    p1[i] = (i % n + 1) % n + i / n * n;
    p2[i] = (i / n + 1) % m * n + i % n;
    p3[i] = (m - 1 - i / n) * n + (n - 1 - i % n);
  }

  set<Permutation> s;
  for (int i1 = 0; i1 < n; i1++) {
    for (int i2 = 0; i2 < m; i2++) {
      for (int i3 = 0; i3 < 2; i3++) {
        s.insert(p);
        p *= p3;
      }
      p *= p2;
    }
    p *= p1;
  }

  int sum = 0;
  for (Permutation const &p : s) {
    sum +=
      1 << countCycles(p); // 2 ^ countCycles(p)
  }
  return sum / s.size();
}
```

Như vậy, qua bổ đề Burnside và định lý liệt kê Pólya, ta có thể rút ra bài học rằng, thằng nói trò đùa của bạn to hơn bạn sẽ nổi tiếng hơn bạn.

Đọc thêm

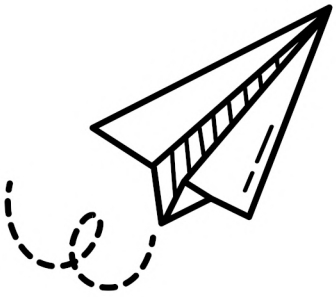
Tham khảo:

- cp-algorithms [100]
- Brilliant [101]
- almostsurelymath [102]

[100] https://cp-algorithms.com/combinatorics/burnside.html
[101] https://brilliant.org/wiki/burnsides-lemma/
[102] https://almostsurelymath.blog/2019/09/18/necklaces-and-groups-the-burnside-lemma/

Phỏng vấn Nguyễn Xuân Tùng

Interviewer: Vương Hoàng Long – ICPC World Finalist 2021



Long: Chào thần Mèo! Cảm ơn thần Mèo đã nhận lời tham dự phỏng vấn của VNOI! Theo đúng truyền thống, thần Mèo có muốn giới thiệu về bản thân mình không?

Tùng: Hỏi khó vãi... "Chào các bạn, mình là Mèo", giới thiệu xong rồi đấy!

Long: Bạn có thể giới thiệu qua một chút về tên tuổi, nơi làm việc và các thành tích không?

Tùng: Mình là Nguyễn Xuân Tùng, hay có một tên "thật" khác là Mèo. Hiện tại mình là sinh viên "đang chờ bị đuổi học" của trường Đại học Quốc tế - ĐHQG TP.HCM. Mình tự hào có được một thành tích mà ngay cả anh Phạm Văn Hạnh, người có 2 cúp vàng Olympic Sinh Viên cũng chưa có được, đó là 2 cúp bạc liên tiếp. Mình khá tin là thành tích của anh Hạnh đã có một người khác đạt được rồi, còn thành tích của mình thì chưa ai làm được cả.

Long: Hai năm liên tiếp về vị trí thứ hai, một thành tích vô tiền khoáng hậu. Nó thể hiện tấm lòng bao dung cao cả của Mèo, nhường cúp vô địch cho các em. Hai năm đều là nhường cho các em, mà hình như đều là hai em năm nhất?

Tùng: Vâng, đều là hai em năm nhất.

Long: Được biết Mèo là một người cực kỳ khủng về CP[103] , một trong số rất ít rating 2600+ Codeforces[104] . Nhưng Mèo lại không có giải quốc gia ở cấp 3, và năm 1, năm 2 ở đại học thì cũng rất là yếu. Để câu chuyện thêm phần kịch tính và để lại những phần hay nhất ở cuối cùng, mình sẽ bắt đầu từ cấp 3. Thế thì, cấp 3 Mèo học ở đâu, học chuyên gì?

Tùng: Cấp 3 mình là học sinh trường Phổ thông Năng khiếu ở TP.HCM, và mình học theo chuyên Sinh. Mà thực ra ngay cả câu chuyện mình vào chuyên Sinh như thế nào cũng là một câu chuyện thú vị nhá! Hồi đấy thực ra mình định vào chuyên Hoá cơ, nhưng mà lúc mình mua đề ôn để thi vào cấp 3, mình có thử đọc hết tất cả các đề của Năng khiếu. Lúc mình đọc tới đề của bên Sinh mình thấy có vài câu làm được. Thế là đúng cái hôm đăng ký thi vào lớp 10, mình đã quyết định đánh thêm môn Sinh vào. Cuối cùng thế nào lại trượt hết tất cả các chuyên, chỉ đậu mỗi chuyên Sinh.

Long: Và cuối cùng thì bạn đã học chuyên Sinh?

Tùng: Thật sự là mình vào chuyên Sinh 3 năm cấp 3, khổ sở nhất là môn Sinh khi mà mình đi ôn thi gần như tất cả các chuyên, chỉ trừ Toán - Tin và môn Sinh thôi. Nên tự dưng đậu chuyên Sinh nên là học nó cũng có những cái khó khăn nhất định.

Long: Thế thì, khi mà học chuyên Sinh, Mèo có thi đội tuyển cấp trường, quốc gia các thứ không?

Tùng: Cái đấy thì không, và thực tế hồi học cấp 3 Mèo luôn có điểm trung bình đứng hạng thứ 35/36 lớp.

Long: Thế cũng dành vị trí cuối cùng cho bạn khác đúng không, giống Siêu Cúp ấy?

Tùng: Đúng rồi!

Long: Thật là cao cả! Không bao giờ trở thành người đặc biệt nhất. 3 năm cấp 3 trôi qua với Mèo học chuyên Sinh, sau đó đến đại học thì Mèo học gì? Đến đại học, Mèo không có giải Quốc Gia tức là Mèo phải thi đại học phải không? Mèo thi đại học khối nào và quá trình đấy như thế nào?

Tùng: Hồi đấy, mình cũng nghĩ nó là một thể loại cơ duyên khá đặc biệt đấy! Bởi vì thường là học cấp 3 chuyên Sinh, bạn chỉ nghĩ được việc vào Y hay Dược hoặc cùng lắm là Công nghệ Sinh học và các ngành tương tự thôi. Lý do ban đầu mà Mèo nhắm đến Công nghệ Thông tin là tại vì Mèo có một đàn anh hơn 1 khoá. Các bạn sẽ biết chắc là bạn nào rồi đấy.

Long: Mình... không biết!

Tùng: Là bạn Lê Duy Bách, cũng là một bạn bây giờ là... Mình ừm....Mình chơi với Lê Duy Bách từ hồi mình lớp 9 rồi, nhưng mà thực ra câu chuyện đấy không quan trọng. Lê Duy Bách hơn mình 1 tuổi. Năm ấy mình lớp 12, lúc đấy Lê Duy Bách cũng mới bước chân vào CP thôi. Ông đấy thách đấu mình một bài trên SPOJ. Bài đấy thật ra input đơn giản lắm, chỉ bao gồm 3 số nguyên và output 1 số nguyên thôi, phần còn lại toàn là Toán học, không phải kiến thức giải thuật gì cao siêu. Nhưng kết quả bài đấy mình đã giải được trước ông Bách, nên lúc đấy mình mới bắt đầu nhắm đến CNTT như một phương án dự phòng. Một phần tại vì thời gian lúc đấy mình ôn thi đại học, vào Y điểm năm nào cũng cao nhất mà, rồi mình cũng xác định là chắc cũng không cạnh tranh lại các bạn khác.

Tùng: Tại sao mình lại vào trường Đại học Quốc tế? Cũng đúng năm đấy là trường Đại học Quốc gia mới bắt đầu có kỳ thi Đánh giá năng lực. Riêng năm của mình thì lại mới chỉ tổ chức cho trường Đại học Quốc tế thôi, mấy năm còn lại mới bắt đầu tổ chức cho toàn bộ Đại học Quốc gia. Năm ấy rốt cuộc thế nào mình thi Đánh giá năng

[103] CP: Competitive Programming - Lập trình thi đấu

[104] Codeforces: một trang web tổ chức các cuộc thi lập trình thi đấu với các dạng đề, bài tập đa dạng.

lực ăn một cái tốp khá cao, và được học bổng.

Long: Thế cuối cùng Mèo vô Đại học Quốc tế là vì tốp cao được học bổng?

Tùng: Dạ đúng vậy.

Long: Thế cuối cùng Mèo có thi Y không?

Tùng: Đương nhiên là có, và cũng đương nhiên là trượt!

Long: Bị trượt nhiều không?

Tùng: Năm ấy, năm 2017, là năm đầu tiên mà chuyển từ thể thức thi trắc nghiệm 90 phút / 50 câu xuống còn 50 phút / 40 câu mỗi môn. Điểm cao khủng khiếp và Mèo không những trượt Y, mà tra điểm mới thấy Mèo đã trượt Y Đa khoa của tất cả các trường Y cả nước. Các nghề khác thì có thể đậu, nhưng mà riêng cái nghề Y thì toàn trượt.

Long: Nói chung điều đấy đã đưa Mèo vào được Đại học Quốc tế, là một trường đại học mà thực tế là không có truyền thống gì về lập trình thi đấu. Mèo bắt đầu với lập trình thi đấu như thế nào, và tại sao tự nhiên lại trở nên mạnh như ngày hôm nay vậy?

Tùng: Câu đấy Mèo chả biết trả lời như thế nào.

Long: Thế năm nhất thì Mèo đã tham gia thi ICPC[105] chưa?

Tùng: Thực ra năm nhất thì chưa. Có điều thú vị này là năm nhất và năm cuối của Mèo ấy, ICPC Việt Nam đều tổ chức ở TP.HCM, và đấy là 2 năm duy nhất mà Mèo không tham gia.

Long: Thế năm nhất của đại học thì Mèo đã bắt đầu tham gia Codeforces chưa?

Tùng: Năm đấy mình không thi ICPC nhưng mà có thi Olympic Tin học, cũng bắt đầu từ khối Không chuyên nên cũng có đánh Codeforces từ thời điểm này rồi. Mặc dù đánh toàn giải được 1 hoặc 2 bài div 2, và 6 contest đầu tiên toàn ăn trừ thôi.

Long: Thế thì Mèo thi Không chuyên năm đấy có giải không? Thực ra Không chuyên thì chắc là vẫn có giải thôi.

Tùng: Vẫn có được giải Ba, năm đấy Mèo được điểm khá cao ở bài rất khó trong khi có một bài khác rất dễ tất cả mọi người đều được 27/30 điểm thì Mèo chỉ có được 10/30 điểm.

Long: Mình đoán là Mèo cũng nhường các bạn khác như mọi khi đúng không?

Tùng: Nhường kiểu này hơi quá! Chưa biết nhường như nào nên nhường hơi nhiều, sau này nhường ít lại .

Long: Thế thì đến năm 2, mọi việc đã thay đổi như nào?

Tùng: Đến năm 2, lần đầu tiên mình đánh ICPC, phải nói là thật sự choáng ngợp với khả năng của các bạn. Năm đấy là một năm mà đề ICPC cũng tương đối khó, thời đề vẫn còn khó hơn bây giờ. Và mình cũng hơi bị choáng ngợp về việc tại sao lại có nhiều đội "xài" được nhiều bài như thế, thậm chí là các đội cấp 3, các bạn năm nhất các thứ. Nói thật, nhìn sức mạnh của các bạn mà mình không nghĩ được là... Nói chung là nhìn lên top trên của mình, mình cảm giác là một cái đẳng cấp mà mình không bao giờ chạm được. Và có vẻ như với số lượng cúp vô địch mình có bây giờ thì đúng là nó là một đẳng cấp mà cũng chưa chạm tới thật .

Long: Năm 2 của Mèo là năm bao nhiêu trong thực tế rồi?

Tùng: Năm đấy là 2018, ICPC tổ chức ở Hà Nội. Thực ra đấy là năm đầu tiên mà Mèo được một cái cúp giải nhì. Đấy là thi Olympic Tin học Sinh viên của Chuyên tin. Năm ấy Mèo được hạng nhì, hạng ba cũng là một bạn khá quen thuộc với VNOI - đó chính là bạn Đỗ Đình Đắc, và hạng đầu tiên chính là bạn đã giới thiệu mình vào CP (Lê Duy Bách).

Long: Ồ ok. Các huyền thoại đều bước ra từ khối Chuyên tin, không có huyền thoại nào bước ra từ khối Siêu cúp cả. Mình đang nhìn cái bảng rating Codeforces của Mèo vào năm 2018. Đúng là từ cuối 2017 đến tầm tháng 12/2018, Mèo có sự tăng rating rất ổn định và gần như contest nào cũng thi, thi rất nhiều luôn. Và đến tầm tháng 12/2018 là Mèo đã có rating gần 2100 rồi, từ đầu năm 2018 rating của Mèo là 1579 rồi đến cuối năm thì đã là 2100. Thì bí kíp Mèo tăng rating một cách quá khủng khiếp trong một năm là như thế nào?

Tùng: Thực ra mình nghĩ bí kíp lớn nhất của mình lại là mình đánh rất chill. Một phần cũng là mình đánh Codeforces cho vui. Thứ hai nữa là động lực đánh chính của mình là những người mà hơn rating của mình một tí thôi. Ví dụ như động lực của mình lúc đấy là một số người quen, như khi mình màu nick xanh lơ chẳng hạn, mình có biết một số người nick xanh đậm. Rõ ràng những người đấy là mạnh hơn mình rồi. Nhưng mà chỉ cần có 1 contest mà mình hơn được người ta thôi là mình cảm giác như kiểu: "Hoá ra mình có thể đánh lại được".

Long: Nói chung bí kíp ở đây đó là cạnh tranh với cả mọi người, đúng không?

Tùng: Vâng, lúc đấy có thể mọi người mạnh

[105] ICPC: International Collegiate Programming Contest - Cuộc thi Lập trình Quốc tế lâu đời và danh giá nhất dành cho sinh viên các trường đại học và cao đẳng trên toàn cầu.

hơn. Nhưng mà riêng việc chỉ cần 1 contest mình hơn được người ta thôi là nó cũng tăng được sự tự tin nhiều rồi.

Long: Thế trong năm 2018 đấy thì Mèo train từ những nguồn nào mà có thể tăng rating kinh khủng như này?

Tùng: Thực ra từ năm đấy mình đã đăng ký tài khoản ở hầu hết các online judge rồi. Nhưng có 2 cái series contest mình thích nhất, một là các contest Codeforces, tại các contest này nhìn chung là vừa diễn ra đều đặn mà vừa chất lượng khá cao. Cái thứ hai mình muốn nói đến là contest 10 ngày của Codechef (contest Codechef Long). Hình như bây giờ thì series contest này không còn nữa và nó được thay bằng một cái rated dành cho div 3, div 4 và hình như chỉ kéo dài 3 ngày thôi thì phải? Nhưng mà các contest Codechef Long là các contest mà mình nhận ra khả năng của mình có thể tiến xa hơn. Dùng cả 10 ngày đấy, mình đã phải research suy nghĩ ra các cách giải khác nhau, các thuật toán mới. Và mình đã cố gắng, thậm chí là đến cả 1 tuần, có thể giải được những bài mà đúng ra là trình độ lúc đấy của mình là khó không tưởng..

Long: Mình nghĩ phương pháp mà giải một bài khó so với trình độ của mình để vượt lên là một cách chắc cũng tương đối nhiều người dùng, và nó khá hiệu quả. Thế năm 2 Mèo thi ICPC ở Hà Nội đúng không? Thế cuối cùng năm đấy Mèo rank bao nhiêu?

Tùng: Năm ấy Mèo giải được 2 bài nhưng mà rank thì hình như là rank xấp xỉ 29, 30 gì đấy.

Long: Năm 2018 thì Mèo train Codeforces, lúc đấy chiến lược train, ví dụ như Codechef Long kiểu dành cả ngày để làm các bài rất khó rồi thì không tính. Thế các bài Codeforces thì Mèo làm theo kiểu virtual xong rồi đọc solution hay là thế nào?

Tùng: Thường là mình đánh contest chính thức sau đó đọc solution, hoặc một bài ngay sau đấy thôi, hoặc là bất cứ bài nào mình đọc và cảm thấy thú vị.

Long: Ngoài ra thì Mèo có làm các contest cũ không hay là chỉ có đánh contest rồi xong đọc solution các bài sau đấy thôi?

Tùng: Mình thường khá ít khi đánh các contest cũ.

Long: Khá hay. Thực ra chính vì thế nên nếu các bạn vào nhìn bảng rating Codeforces của neko_nyaaaaaaaaaaaaaaaa sẽ thấy, năm 2018 là gần như contest nào cũng thi, thi rất nhiều luôn. Vì thi nhiều như vậy nên có rất nhiều bài để làm, không cần đánh contest cũ. Thế năm 2018 thì Mèo có giải nhì khối Chuyên tin nhỉ... à đâu rank 2 khối chuyên Tin.

Tùng: Rank 2 thôi, chuyên Tin năm nào cũng nhiều hơn 1 giải nhất.

Long: Thế là rank 2 khối chuyên Tin, đứng trên một huyền thoại khác đó là Đỗ Đình Đắc và đứng dưới một siêu huyền thoại khác. Trở về năm 2019, Mèo thi rất ít luôn, chiến lược gì đã thay đổi? Đầu năm thì Mèo rating khoảng 2100, và cuối năm thì Mèo rating là đỏ Codeforces. Đúng rồi, đỏ Codeforces 2423, lần đầu tiên Mèo lên đỏ Codeforces là tháng 12 năm 2019. Nhưng mà Mèo làm ít contest hơn hẳn thì chiến lược đã thay đổi như nào?

Tùng: Năm đấy vẫn còn những cuộc thi dài Codechef Long, mình có một cái cộng đồng chuyên đi làm Codechef Long. Thật ra mình không nhớ lắm, năm 2019 là khoảng thời gian mà mình bắt đầu làm Codechef Long nhiều hơn và tập các bài khó nhiều hơn.

Tùng: Với lại thật ra kiểu training của mình là xuất thân từ thi ICPC chứ không có thi VOI, nên lúc đầu lên rating nhanh một phần cũng vì thi ICPC bạn có thể chép thuật toán. Ví dụ như kiểu dùng thuật toán như một cái hộp đen ấy, mà thậm chí có một số bài toán còn không cần hiểu thuật toán đấy. Đương nhiên là hiểu thì vẫn giải được nhiều bài hơn, nhưng mà thực tế là bạn cũng có thể tiến được một khoảng cách nhất định nếu như bạn biết thuật toán làm gì thôi.

Tùng: Mình nghĩ giai đoạn này là giai đoạn mà mình chậm lại và bắt đầu học kỹ hơn. Hoặc cũng có thể đơn giản là lúc đấy mình chỉ muốn nghỉ ngơi thôi và tương tự như vậy. Hoặc là mình đổi chiến lượng training sang luyện tập thay vì là đánh contest chẳng hạn? Ví dụ bạn có thể thấy là giai đoạn cuối năm 2019 mình đánh contest nhiều thế nào. Và đặc biệt như bạn nói là cuối năm 2018 và cuối năm 2019 Codeforces mình lên một màu mới, nhưng thực tế toàn là lên sau đợt Olympic Tin học Sinh viên.

Long: Àaaa, thú vị. Tức là sau khi đạt giải Olympic Sinh viên thì Mèo tự thấy là sức mạnh của mình tăng lên và vào thi Codeforces tăng lên thật đúng không?

Tùng: Cái đấy... mình nghĩ là chỉ có trời mới biết thôi chứ mình cũng không biết đâu!

Long: Thế Mèo năm 2019 thi ICPC thì kết quả như nào và vẫn thi bảng Chuyên Tin à, hay là thi theo bảng Siêu Cúp?

Tùng: Năm đấy là mình thi bảng Siêu cúp rồi, bảng Không chuyên với Chuyên tin là có giải ba trở lên là năm sau sẽ không được thi bảng đấy nữa.

Long: Thế thì Mèo thi bảng Siêu Cúp thì kết quả như thế nào?

Tùng: Năm đấy là năm đầu tiên mình thi Siêu Cúp, và kết quả là mình... không được giải gì hết.

Long: Ừm tại vì... thực ra thì Siêu Cúp rất khó có giải. Nó chỉ có 6 giải thôi, không có đến mấy chục giải như bảng Chuyên Tin với cả Không chuyên. Thế thì còn ICPC?

Tùng: ICPC năm đấy thì mình vẫn trên tinh thần là để gặp mọi người nhưng thực ra mình làm khá tốt so với mong đợi. Năm ấy chính xác là mình được hạng 18. Đấy là mình đã miss một bài dễ nhưng mà đã giải được 2 bài khó. Mình mà giải thêm được bài dễ đấy thì mình nghĩ mình cũng có thể tự tin nói rằng mình nằm trong tốp đầu.

Long: Không rank 18 thực ra là khá tốt ấy chứ. Khi xem bảng Official thì team Mèo sẽ có huy chương đúng không?

Tùng: Đúng, có huy chương đồng.

Long: Ah nice, các bạn có thể thấy là Mèo có sự đi lên rất là rõ ràng luôn, mỗi năm lên một màu và mỗi năm đi lên một kiểu. Nói chung, năm 2019 này Mèo tập trung vào train Codechef Long. Và cuối cùng thì hết năm 2019, bạn đạt được một thành tích, mà ở Việt Nam ngày xưa, từ trước khi COVID diễn ra, là rất ít người có, đó là đỏ Codeforces. Hồi đó đỏ Codeforces Việt Nam đếm được trên đầu ngón tay, và cái đỏ Codeforces ấy rất là quan trọng. Bước sang năm 2020, là một năm mà Mèo suýt lên được 2k6 Codeforces, tức là suýt lên được IGM. Và năm này thì Mèo còn train ít contest hơn cả năm 2019 nữa. Chiến lược của Mèo trong năm đấy là gì?

Tùng: Năm 2020 là lúc đấy mình đang năm cuối đại học. Mình bận học với cả mình bận luận văn các thứ, làm sao mà train nhiều được .

Long: Thế tại sao Mèo train ít mà đến cuối năm nó vẫn gần như lên được IGM như thế này?

Tùng: À thì lúc đấy vẫn giải bài thường xuyên thôi! Thật ra mình kể cả ngay đến tận bây giờ nhá, kể cả những contest mà mình không làm, nhưng mình biết có contest thì mình vẫn luôn lấy đề ra để đọc. Đọc để nghĩ xem mình có giải được không, có chứng minh được không ấy.

Long: Đọc đề trên Codeforces à? Hay là có Codechef Long không? Năm này còn Codechef Long không?

Tùng: Thường là chỉ Codeforces thôi, tại nếu như mình mà chỉ đọc đề ấy thì mình chỉ thường theo dõi thôi, thường theo dõi thì mình chỉ chọn ra những thông tin nào thú vị để đọc thôi. Ví dụ như bình thường, mỗi khi mình theo dõi một contest Div 2 chẳng hạn, thường toàn chỉ đọc từ bài E trở lên trừ khi nào mà có người nào đấy cần, trừ khi mà có ai khác để discuss bài dễ hơn.

Long: Và năm này thì ICPC diễn ra ở Cần Thơ thì phải?

Tùng: Yea, Cần Thơ.

Long: OK, năm này Mèo vẫn tiếp tục thi ICPC?

Tùng: Năm này đương nhiên là có! Và năm 2020, ICPC với mình là một kết cục có thể nói là bi thảm! Không phải vì làm thọt mà vì làm quá bay nhưng lại không được đi WF[106] ;(

Long: Là rank bao nhiêu nhỉ?

Tùng: Năm đấy mình RANK 4. Và kết quả là đội TOP 3 ĐƯỢC ĐI WF. Đội ở rank 3 đến từ trường HCMUS, có bạn Nguyễn Diệp Xuân Quang. Mình đã thua đội bạn đấy chỉ 24 phút penalty.

Long: 24 phút là quá nhiều. Thật là cay đắng khi thua penalty. OK, thế còn bảng Siêu Cúp, năm này kết quả Mèo thế nào?

Tùng: Riêng bảng Siêu Cúp năm đấy, mình thọt nhưng mình lại là thí sinh đứng bét có đồng giải 3. Thực ra Siêu Cúp thường là trao 10 giải hoặc nhiều hơn. 6 giải thì là có cúp này, các giải còn lại thì gọi là đồng giải 3. Giống như giải Khuyến khích ấy, mặc dù nếu giải KK kiểu này thì cũng khó lấy vãi chưởng ra.

Long: OK. Thế thì kết thúc năm 2020, và đến năm 2021, 2022, thì đúng là thời của thần Mèo, vô địch thiên hạ! Vừa vào 2021 được 28 ngày thì Mèo lên được IGM, và là một trong số rất rất ít IGM của Việt Nam. Nhưng vì Mèo để quốc tịch trên Codeforces là Lào, nên là nó không hiện lên trên cái bảng đó. Thế thì 2021, lần đầu tiên Mèo được qualify WF, hình như là rank 3, thi ở Hà Nội. Chiến lược cho cái năm này Mèo đã train như thế nào?

Tùng: Năm 2021, chiến lược train của mình khá khác! Thật ra thì cũng gần như không có chiến lược train mấy, vì lúc đấy mình trên tinh thần là nếu như năm 2020 mình gần qualify được WF, thì...

Long: ... năm 2021 chắc chắn là qualify cho WF.

Tùng: Đúng! Năm 2021, đúng cái kỳ thi quan trọng là kỳ thi Regional, mình lại khá ngạc nhiên mình làm bay như thế, vì tất cả các contest vòng miền và contest vòng National mình đều thua đội của HUST. Năm đấy ban đầu dự tính là hình như

[106] WF: World Finals – Cuộc thi ICPC cuối cùng trong một mùa giải bao gồm những đội xuất sắc nhất của các trường đại học trên thế giới vượt qua vòng loại vùng (Regional contest)

Việt Nam có 3 suất, mà mấy vòng liền mình gần như chỉ ăn là suất thứ tư. Lúc đấy, mấy cái ICPC này mang tính ganh đua, và việc qualify vào WF nó rất là ganh đua nhá, nên kể cả mình mạnh đến đâu thì vẫn sẽ có người mạnh hơn. Hơn nữa, nói nhỏ thôi chứ nếu như một mình mình giải hết mà đánh với 3 người khác đều có khả năng giải bài, thì nó là một cái... (bật cười) mình cảm giác là một cái gì đấy rất là không tưởng.

Nguyễn Xuân Tùng tại kỳ thi Thách Thức 2022

Long: Đúng rồi, mình cũng phải công nhận là năm đấy Mèo qualify quá ngoạn mục. Nói chung, mọi người không thể biết được, trong phòng ban giám khảo, mọi người đã cảm thấy nó kinh khủng như nào khi mà team Mèo có thể đứng rank 3 mà rank 3 toàn thời gian thi luôn. Không phải kiểu phút cuối ăn may đâu. Năm 2021 thì thực ra Mèo train Codeforces ít hơn tất cả các năm trước nhá, thì chiến lược năm này là gì? Có phải cày Codeforces không, hay là Mèo có tự cày không?

Tùng: Thực ra, từ 2020 trở đi, lý do contest mình ít là tại vì toàn là contest div 1 thôi. Nói thật là hồi đấy mình cũng có tinh thần farm rating một tí, nên là tất cả các contest mình đánh toàn là Div 1 + Div 2.

Long: Ôi thực ra là khá hợp lý. Ê đúng ha!

Tùng: Những contest này thực sự dễ farm rating hơn. Bởi vì thứ nhất là nếu bạn Div 1 thì bạn sẽ được mong đợi là hơn rank của hầu hết các Div 2. Thứ hai nữa là kể cả nếu bạn có làm thọt thì bạn gần như không thể nào thọt được các bài dễ nhất. Nên thường là rating kỳ vọng của bạn sẽ khá cao.

Long: Thế ngoài làm Codeforces thì trong thời gian này Mèo có train gì nữa không?

Tùng: Thực ra, cái khoảng thời gian mình train mạnh lại là tầm từ khoảng giữa đến cuối năm, vì đầu năm là HK2 năm tư của mình, và đúng là mình bận làm luận văn thật. Lúc đấy mình train chủ yếu là hoàn thiện lại các cái điểm còn sót từ năm Cần Thơ trước đấy thôi. Mình thấy mình cũng khá may mắn. Thực ra nói may mắn không đúng tại vì 2 teammate của mình thực ra có đánh CP từ trước rồi, mặc dù kết quả vào ai giải bài nào thì cũng như thế.

Long: Năm 2022 mình cũng sẽ skip qua vì năm 2022 Mèo không thi ICPC nữa và Codeforces thì cũng thi mỗi 4 contest. Nói chung hoạt động train cũng không mạnh nữa vì đã đạt đến đỉnh cao của việc qualify WF lần đầu tiên trong lịch sử trường Đại học Quốc tế. Nhìn chung thì các bạn có thể thấy, Mèo có một quá trình phát triển rất đều và rất khủng khiếp qua các năm: từ đầu năm 2018 với rating đâu đó khoảng 1500, đầu năm 2019 là 2100, đầu năm 2020 thì là 2400, và đầu năm 2021 là 2600. Sự phát triển mà gần như là không mấy ai có được. Khi nhìn lại thì Mèo thấy chiến lược train của mình nó đúng đắn nhất ở chỗ nào?

Tùng: Nói về chiến lược train của bản thân, mình nghĩ là nó cũng có điểm đúng và nó cũng có điểm sai. Mình nghĩ điểm đúng ở chỗ là đầu tiên cái mindset train của mình nó khá chill. Kiểu mục tiêu của mình là chỉ muốn 1 contest mình hơn được những người mình cho là hơn hẳn mình. Mình chỉ cần thắng 1 contest thôi thì tự dưng mình có cảm giác hơn người ta. Và hai nữa là mình, mình nghĩ cái mình mạnh vì mình train nhiều, chứ không hẳn do mình mạnh lắm .

Long: OK, còn gì nữa không?

Tùng: Với cả mình nghĩ cách train của mình nó cũng hợp với ICPC hơn là VOI. Ở thi ICPC thì bạn được quyền mang notebook vào mà, cho nên bạn có thể học được rất nhiều thuật toán mà không nhớ cách cài, cũng có thể là bằng cách đấy mà mình train được rất nhanh. Nhưng mà cũng mất kha khá thời gian mình mới cảm thấy thật sự hiểu về CP hơn.

Long: Thế thì khi train, ngoài những cái

Codechef Long, thì các bài Codeforces, trong những năm mà Mèo không còn làm Codeforces thường xuyên như là năm 2020, 2021, 2022, Mèo có lấy Codeforces cũ ra làm không hay là cũng chill thôi?

Tùng: Thường thì mình hay train chính bằng các cái problemset ngoài hơn là các cái problemset của các online contest.

Long: Thế thì Mèo có recommend problemset nào để mọi người train không?

Tùng: Mèo có một cái thứ mà Mèo luôn luôn recommend cho các bạn, đấy là CSES, problemset của Antti Laaksonen. Ông này cũng có một cuốn sách, cũng là một cuốn CP handbook. Cuốn này vừa có một phiên bản free trên trang web và một phiên bản đầy đủ, cái này bạn phải mua, năm phát hành mới nhất của cuốn này là 2020. Mình luôn recommend problemset này. Và ngoài ra mình có một cách train khá kỳ lạ, đó là mình hay lấy một stream contest nào đó ra và sau đấy cố gắng giải cho bằng hết. Cảm giác khi được train trên một cái problemset thực tế, mình lại có cảm giác hứng thú hơn! Khi đấy nếu như mình vô tình giải được nhiều bài thì mình lại nghĩ nếu mình vào một contest thật, mình sẽ được một thứ hạng rất cao. Nên mình cảm thấy train trên một cái problemset đầy đủ hay train trên một cái problemset onsite nào đấy mà giá trị của contest nó cao một tí, thì sẽ mang lại cảm giác hứng thú hơn.

Long: Khá là thú vị. Thế khi Mèo giải một cái problemset, bài trên CSES chẳng hạn, sẽ cố gắng giải cho tới khi nào giải được thì thôi hay là sau một thời gian nào đấy bạn sẽ dừng để đọc solution?

Tùng: Cái này thú vị, còn tùy tính chất của bài toán nhá. Nếu như một bài mà mình cảm thấy nó thú vị hoặc là mình đã biết đây là một bài mà nó không có thuật toán đặc biệt, thì mình sẽ cố gắng giải. Khi bạn mà ngồi xuống tập trung làm bài và suy luận thì nó có thể ra được những solution một cách khá là ngạc nhiên. Nhưng còn những bài còn lại mà mình biết nó có một thuật toán gì đấy, nếu mình biết thì mình sẽ giải ra, mình không biết thì mình sẽ Google solution để xem nó xài thuật toán gì. Nói chung tuỳ vào mình đã giải được đến đâu. Khi giải một bài toán, bạn sẽ phải đi từng bước, bạn sẽ phải đưa bài toán cho đến khi ra được một cái hình thù gì đấy mà mình nhận ra được. Mình sẽ thường sẽ phải tự quyết định xem là đến giai đoạn nào mình không thể tìm được nữa mà phải đọc đến solution.

Long: OK. Để tổng kết lại cái quá trình train và các thứ thì Mèo có lời khuyên gì đến cho các bạn đang nghe không?

Tùng: Mình nghĩ là có một cái điều mà mình vẫn thường hay nói với mọi người đấy là: "Competitive Programming is about having fun, if you're not having fun then you're doing it wrong." Nếu như mà các bạn train CP thì các bạn nên cảm thấy vui vẻ cơ! CP thực tế chính xác là một môn thể thao. Nó mang tính chất ganh đua, nhưng kết quả cuối cùng vẫn là tất cả mọi người đều phải lên trình độ được. Thực ra hồi đầu train mình cũng không nghĩ là kết quả sánh được với các bạn tốp cao. Có lẽ quan trọng hơn là mình không có đặt nặng vấn đề là phải lên được tốp như thế nào đấy. Thậm chí có khi chỉ cần lên một hạng so với những gì mình mong đợi thôi nó cũng là một bước tiến khá lớn. Khi mình có bước tiến đó rồi, mình mới bước được bước tiếp theo. Chốt lại, lời khuyên của mình có lẽ chỉ có tóm gọn vào việc là hãy chill và vui vẻ, xong rồi đánh contest thôi. Cuộc thi nào cũng sẽ có kẻ thắng kẻ thua cả!

Long: OK, và câu hỏi cuối cùng cũng như mọi khi đó chính là Mèo đã tốt nghiệp chưa nhỉ, hay là bị đuổi chưa nhỉ?

Tùng: Đang chờ bị đuổi, Mèo nộp bằng tiếng Anh rồi, nhưng mà phải chờ đến khi nào trường có đợt đuổi, à nhầm xét tốt nghiệp tiếp theo thì mới được.

Long: À quên! Chúng mình chuẩn bị kết thúc cuộc phỏng vấn, nhưng mà Mèo nhắc đến tiếng Anh nên là thêm 5 10 phút cho các bạn.

Tùng: Ui giờiiiii

Long: Mèo là một người rất là huyền thoại nhá, ngoài việc không học Tin và cũng chả học chuyên Toán, học Đại học Quốc tế và 2k6 Codeforces, thì Mèo còn có IELTS Speaking 8.0 đúng không?

Tùng: Thực ra Speaking em có 8.5 nữa, overall em mới 8.0 thôi.

Long: Mèo nói tiếng Anh hay lắm các bạn ạ! Mèo có thể chia sẻ kinh nghiệm học tiếng Anh trong khi mà vẫn 2k6 Codeforces được không?

Tùng: Kinh nghiệm học tiếng Anh à? Ôi câu này khó, tại mỗi lần người ta hỏi...

Long: Mình vẫn giỏi đúng không?

Tùng: Thôi thôi, kinh nghiệm tiếng Anh thì em chịu chết.

Long: Kinh nghiệm Speaking là bạn phải giỏi sẵn, không có học gì lên 8.5 đâu.

Tùng: Về việc học Tiếng Anh, lời khuyên mình không có nhưng mà lời khuyên về việc thi IELTS thì... Mình nghĩ thi IELTS ngoài việc mục đích đo trình độ của mình thì cũng là để show off.

Thi IELTS mình phải show off được cái khả

năng của mình. Mình biết bao nhiêu ngữ pháp lôi hết ra, bao nhiêu từ vựng lôi hết ra. Nói chung là mình hãy cứ trực tiếp show off cái khả năng của mình, khi đấy điểm có thể sẽ cao lên. Cái việc show off mà điểm cao lên, mình cũng không nghĩ là một điều xấu đâu nhé! Vì khi mình làm như thế, ban giám khảo mới biết được là mình mạnh thế nào. Còn nếu như mình mạnh mà người ta không biết thì người ta khó chấm điểm lắm. Bởi vì IELTS này nó là một cuộc chơi, không chỉ riêng mình mà của mình và ban giám khảo nữa, trừ 2 môn mà có sẵn đáp án thì không nói làm gì

Long: OK, một kinh nghiệm rất là hay ho, các bạn nhớ show off khi đi thi IELTS để được 8.5 Speaking như Mèo - một thành tích mà mình tin là rất nhiều giáo viên tiếng Anh cũng mơ ước luôn chứ không chỉ là các bạn bình thường. OK! Để kết thúc cuộc phỏng vấn giống như tất cả các khách mời khác. Trong tương lai, sau khi bị đuổi hay tốt nghiệp, Mèo có dự định gì trong 5 năm, 10 năm?

Tùng: Phỏng vấn mà 5 năm, 10 năm thì khó quá.

Long: Không 5 năm, 10 năm thì 1 năm thôi, thần Mèo là người đặc biệt mà nên là dự định 1 năm thôi, hoặc 1 tháng thôi cũng được.

Tùng: Mình không hay tính xa lắm, thực ra cái mình tính xa nhất đến giờ là WF năm sau thôi. Kể cả WF năm sau thì mình chắc không mạnh như team EggCentroy được đâu! Nhưng mình đặt mục tiêu trong WF là làm một cái gì đấy để mình không hối tiếc. Ví dụ như đạt một cái rank trong tốp 50. hay một cái tốp gì đấy cao cao hơn mình mong đợi chẳng hạn. Hoặc là giải được một cái bài khó, first solve được bài dễ nhất. Nói chung là làm cái gì đấy để lại dấu ấn đặc biệt một chút, giống như 2 cúp bạc siêu cúp mình đã đạt được .

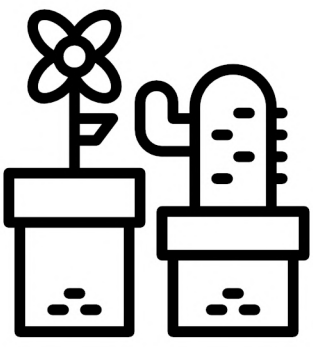
Long: Yea thế thì chúc Mèo may mắn trong WF năm tới! Biết đâu Mèo lại làm một pha, ví dụ như, rank 2 WF chẳng hạn.

Tùng: Ui giồi ơiiiiii .

Long: Đi vào lịch sử luôn, EggCentroy ra đường . OK thế thôi, cảm ơn Mèo đã nhận lời phỏng vấn ngày hôm nay. Chúc Mèo năm mới, Tết vui vẻ, thế thôi! Chào bạn nhé!

Phỏng vấn Lê Bảo Hiệp

Interviewer: Mai Vinh Hiển – Tình nguyện viên VNOI Gen 2



Hiển: Xin chào Hiệp, bạn có thể giới thiệu đôi chút về bản thân, công việc hiện tại cũng như thành tích nổi bật của bạn được không?

Hiệp: Mình là Lê Bảo Hiệp, sinh viên năm 3 của Chương trình Tiên Tiến ngành Khoa học máy tính, trường Đại học Khoa học Tự nhiên. Hiện tại mình cũng đang thất nghiệp . Ở cấp 3 mình có Giải nhì Học sinh giỏi Quốc gia môn Tin học và vào vòng 2, cơ mà không được dự thi tiếp vòng Châu Á hay Quốc tế nào cả . Sau đó lên Đại học thì mình được 2 lần vô địch ICPC[107] Regional và cũng có cho mình Cúp Đồng khối Siêu cúp Olympic Sinh Viên.

Lê Bảo Hiệp và team Burned Tomatoes

Hiển: Wow, các thành tích đều khá là khủng. Theo mình biết thì tới lớp 12 Hiệp mới bắt đầu tham gia đội tuyển và sau kỳ thi Học sinh giỏi Quốc gia, Hiệp cũng là người có thành tích cao nhất đội tuyển. Vậy Hiệp có thể chia sẻ cho các bạn kinh nghiệm học CP[108] của mình và bí quyết để đạt những thành tích khủng như thế khi mới bắt đầu vào đội tuyển ở năm lớp 12?

Hiệp: Thật ra hồi cấp 2 thì mình có học ở ngoài Quảng Nam, lên cấp 3 thì mình thi vào trường Phổ thông Năng khiếu. Lúc đấy thì mình học lớp Chuyên Toán chứ không phải lớp chuyên Tin, thì năm 11 mình có thi vào đội tuyển và mình tạch. Vào thời điểm ấy đi thi mình còn sợ một cái đấy là trong lịch sử của Năng khiếu, đội tuyển Tin chỉ có học sinh lớp chuyên Tin thi vào được và mình là học sinh chuyên Toán nên mình có tâm lý sợ là "không biết thầy có dìm mình không" . Do đó lên năm 12 mình đã đắn đo rất nhiều về việc đi thi tiếp, thậm chí có câu chuyện vui là hè năm 11 lên 12 mình mới bắt đầu train lại, tức là nguyên học kì 2 năm 11 sau khi mình thi trượt thì mình không hề cày thêm CP . Giai đoạn đó chủ yếu là mình cày Codeforces[109] , bài trên VNOI và đọc cuốn CP3 của thầy Halim. Thời điểm mình thi vào đội tuyển thì cũng mới rating Expert trên CF thôi nhưng cuối cùng mình vẫn đậu đội tuyển. Trong thời gian học đội tuyển thì chủ yếu mình cày những bài khá là cổ điển, kiểu mấy bài của thầy Lê Minh Hoàng ý. Việc làm những bài cơ bản như thế có một điều lợi là giúp cho kiến thức cơ bản của mình cực kỳ tốt, cho nên lúc đi thi Quốc gia mình có chơi chiến thuật là code vét và code chắc hơn là code full.

Hiệp: Năm đấy đội tuyển của mình người mạnh nhất không phải là mình mà là Nguyễn Vũ Đăng Huy (Hollowed) và sau này cũng là teammate ICPC của mình . Câu chuyện năm đấy là ở Vòng 1 thì Huy có làm full các kiểu còn mình thì chỉ code vét thôi, lúc biết kết quả thì mình đậu Vòng 2 còn nó thì tạch

Hiển: Khá là thú vị. Không biết là hồi đấy Hiệp có học dự tuyển hay trại hè vào lúc lớp 10, 11 không?

Hiệp: À không, mình hoàn toàn không có học chỗ nào luôn , đúng nghĩa là lúc đấy mình tự học hết mọi thứ.

Hiển: Vậy Hiệp lúc đấy có liên hệ với các bạn trong đội tuyển hay các bạn lớp Tin để xin tài liệu hay tips học không nhỉ hay cũng tự mò hết mọi thứ?

Hiệp: Lúc đấy gần như là tự mò, thật ra hồi đấy mình có xin theo học thầy dạy Tin lớp 10 của mình, nhưng được một thời gian mình cảm thấy cũng không ổn lắm nên sau đó đa số mình cũng phải tự học.

Hiển: Theo mình biết, Hiệp là một người code rất khỏe và trâu, đặc biệt khi đã vô địch Regional 2 lần. Vậy Hiệp có thể chia sẻ cách luyện tập để code chắc ở các kì thi, chúng ta có cần phải tập trung vào kỹ năng code nhiều hơn là đầu tư thời gian vào học những thuật toán cao siêu hay phức tạp như kỳ thi HSGQG không?

Hiệp: Đây là kinh nghiệm của mình đối với đề Quốc gia từ năm 2020 đổ về trước nha, những năm sau này thì có vẻ nó không còn đúng lắm . Bình thường bài 1 của cả 2 ngày sẽ là những bài bắt buộc phải ăn full nếu muốn có giải cao, những bài đấy thường khá là dễ nếu như mà học nghiêm túc và học chắc các kiến thức như Quy hoạch động hay Cấu trúc dữ liệu, còn những bài sau, như bài 2 và 3 năm mình thì nó là một bài Hình và một bài Đồ thị, mình năm đó thì cũng vét thôi chứ cũng không nghĩ nhiều gì hết. Về việc training thì như mình có

[107] ICPC: International Collegiate Programming Contest - Cuộc thi Lập trình Quốc tế lâu đời và danh giá nhất dành cho sinh viên các trường đại học và cao đẳng trên toàn cầu.

[108] CP: Competitive Programming - Lập trình thi đấu

[109] Codeforces: một trang web tổ chức các cuộc thi lập trình thi đấu với các dạng đề, bài tập đa dạng.

nói ở phần trước thì cứ học hết những cái cơ bản và cài đi cài lại nhiều lần, như Quy hoạch động, Cấu trúc dữ liệu kiểu Segment Tree, rồi DnC, đồ thị thì là những thuật BFS hay Dijkstra thì mình cũng làm rất là nhiều lần rồi, thậm chí mình có thể gõ mà không cần nhìn nữa .

Hiển: Trong thời gian học đội tuyển thì Hiệp có đặt mục tiêu cho mình về giải hay như đi Vòng 2, IOI[110] không?

Hiệp: Mục tiêu lớn nhất của mình khi đó chắc chắn là Vòng 2 (TST[111]) rồi, vì mình cứ nghĩ tới viễn cảnh đi học đội tuyển về xong thì phải ôn thi đại học nên là lúc đó khi mình học là trong tâm thế "không còn gì để mất" và mình đã học rất là trâu, nhiều lúc bây giờ mình cũng ước là mình có thể học trâu như hồi đấy . Cơ mà thật ra mục tiêu của mình là Vòng 2 hay APIO[112] nếu có thể, còn mục tiêu mình gáy với các bạn khác là mình muốn đi IOI để đè Bùi Hồng Đức các kiểu . Tại mình hay nghĩ theo cái kiểu mà, cứ đặt mục tiêu cao lên thì nếu mình không đạt được mục tiêu đấy thì mình cũng có thể đạt được mục tiêu thấp hơn một tí

Hiển: Hồi đấy khi Hiệp học đội tuyển lớp 12 thì Hiệp cũng thi ICPC khối THPT đúng không? Vậy Hiệp có thể chia sẻ cho mọi người biết về quá trình thi ICPC lúc đấy được không?

Hiệp: Đúng rồi, nhưng mà đợt đấy do là lần đầu tiên thi nên team code rất là thọt, đúng nghĩa hồi đấy thi là chả có chiến thuật nào cả, cứ đọc đề xong biết làm bài nào thì làm, mình còn nhớ đợt đấy vòng National thì team mình chỉ được 3-4 bài gì đấy

Hiển: Sau khi ra trường, Hiệp có dự định về Phổ thông Năng khiếu để dạy và truyền lửa cho các khóa sau không?

Hiệp: Hồi trước mình cũng có được thầy mời nhưng tới giờ thì mình vẫn chưa nhận được thêm thông tin nào cả nên tới giờ vẫn chưa về dạy

Hiển: Ngoài Lập trình thi đấu, Hiệp còn có sở thích nào nữa không?

Hiệp: Hồi đấy Năng khiếu có bóng bàn nên mình cũng chơi bóng bàn khá là nhiều, và thật ra hè năm 11 mình không chơi CP bởi vì mải chơi bóng bàn, kiểu mình lúc đấy chả biết làm gì khác ý . Cơ mà sau này lên Đại học mình cũng ít đánh bóng bàn bởi vì một phần hồi đấy ở trường vẫn còn có bàn sẵn ở đó và người quen để đánh cùng chứ lên Đại học thì cũng ít quen ai với không có bàn sẵn nữa

Hiển: Theo mình được biết thì Hiệp ngoài khủng trong CP thì cũng khủng dev, vậy Hiệp có thể chia sẻ cách học code những cái khác ngoài CP ra được không?

Hiệp: Thật ra hồi lớp 2 mình đã ngồi vọc máy tính các kiểu rồi xong đến năm cấp 2 thì cũng ngồi làm mấy cái phần mềm vớ vẩn bằng Visual Basic để đi thi cuộc thi Sáng tạo thanh thiếu niên nhi đồng của Tỉnh, chả biết bây giờ còn bạn nào biết không . Sau này lên cấp 3 khi mình học CP thì mới có ý định nghiêm túc theo Công nghệ thông tin, thì mình nghĩ cái quan trọng nhất để học được Công nghệ thông tin chính là Tiếng Anh phải rất tốt để đọc các tài liệu nước ngoài, rồi còn lại chủ yếu là tự tìm tòi thôi, tại chủ yếu mình có khá nhiều thời gian rảnh nên là mình cứ lên Internet chứ cũng chả có bí kíp kinh khủng nào cả .

Hiển: Hiệp hiện tại đang làm cho team kỹ thuật của VNOI, cụ thể là VNOJ, vậy ngoài cải thiện khả năng dev ra thì Hiệp cảm thấy mình nhận được thêm những điều mới nào khi làm trong VNOI không?

Hiệp: Mình nghĩ cái lợi lớn nhất khi mình làm cho VNOI là các mối quan hệ, mình được quen biết rất nhiều bạn bè và những người giỏi, để sau này nếu có đi xin thực tập hay việc làm ở Google, Facebook các kiểu thì mình nghĩ cũng rất thuận lợi khi mà có thể có một người nào đó trong VNOI đang làm ở đó xin cho mình refer

Hiển: Theo mình được biết thì những bạn học ở Phổ Thông Năng Khiếu thường sẽ hay có dự định gap year một trường ở Việt Nam hoặc đi du học ngay sau khi tốt nghiệp cấp 3. Tại sao Hiệp lại không chọn đi du học?

Hiệp: Thật ra việc mình vào Khoa học Tự nhiên cũng có thể xem là một thất bại đối với mình . Do hồi đó học ở Năng khiếu thì các bạn mình đi du học ở Mỹ hay các nước rất là nhiều, hồi đó mình cũng có đặt mục tiêu đi Mỹ và thi SAT, IELTS các kiểu nhưng khi tới lúc viết bài luận thì mình khá là nản nên đến cuối mình quyết định học ở Việt Nam. Quyết định mình học APCS ở trường Khoa học Tự nhiên cũng đơn giản là khi nhìn tên các ngành thì đấy là ngành duy nhất ghi tên ngành là Khoa học Máy Tính nên mình chọn chứ mình cũng chả biết chương trình đấy là gì và có những ai khủng học ở đây .

[110]IOI: Kỳ thi Olympic Tin học Quốc tế.

[111]TST: Kỳ thi tuyển chọn đội tuyển dự thi Olympic Tin học Quốc tế, hay còn được biết đến là Vòng 2 thi chọn Học sinh giỏi Quốc gia.

[112]APIO: Kỳ thi Olympic Tin học Châu Á -– Thái Bình Dương.

Hiệp: Cũng khá là may khi mà lúc vào học rồi thì mình mới biết là mình đã chọn đúng, và nếu nhìn theo một cách khác thì ở Việt Nam mình mới có cơ hội đi World Finals[113] chứ qua bên Mỹ thì mình phải vào trường mạnh và mình phải cực kỳ mạnh thì mới được đi, còn vào trường yếu về ICPC thì cũng không có teammate để đi thi cùng

Hiển: Khi lên Đại học thì có vẻ Hiệp không còn nhiều thời gian cho CP nữa, vậy Hiệp đã có cách cày CP nào khác không?

Hiệp: Lên Đại học mình cũng không có cày nhiều, thật ra mình thấy so với trình của mình ở Đại học với trình của mình lúc học thi Quốc gia thì nó cũng không chênh nhau quá nhiều, chắc có một cái khác là mình cài bài trâu và chắc hơn hồi đó thôi chứ não thì chắc cũng chả to lên mấy . Tại theo mình nghĩ mình phải tập trung rất cao độ và thường xuyên cày thì mới lên trình được chứ lên Đại học thì có nhiều môn và khiến mình bị phân tâm nên là cày nhiều lúc nó cũng không hiệu quả lắm.

Hiển: Vậy so với khi thi ICPC ở cấp 3 thì thi ICPC trên Đại học, Hiệp đã trang bị cho mình những chiến thuật nào khi đi thi, như vai trò của Hiệp trong và team sẽ chia ra đội hình là 1 người code 2 người nghĩ hay cả 3 có vai trò giống nhau?

Hiệp: Thật ra team mình thì sẽ có 2 người code chính, thường thì mình sẽ code chính hơn do Phát thì não to hơn và sẽ ưu tiên cho Phát nghĩ bài, còn Huy thì sẽ nghĩ bài luôn và mình cũng không hay cho Huy code, bởi vì Huy code chậm và hay bug . Còn chiến thuật đọc bài thì bọn mình cứ chia đều cho 3 người đọc, như mình thì mình khi đọc đề sẽ đọc phần Output trước tại đề của Việt Nam lúc nào cũng sẽ có một bài cho điểm, có các dạng như kiểu in ra một số hay là YES/NO, như bài G Regional vừa rồi ý, bài đấy thì mình đọc Output xong rồi đọc dần lên rồi code luôn

Hiển: Hiệp có thể chia sẻ quá trình team Hiệp luyện tập cho các kì ICPC được không?

Hiệp: Team mình train khá ít, chỉ có năm ngoái căng hơn một chút khi mà có team anh Nhật thì bọn mình cũng bị áp lực. Thật ra chỉ khi trước Regional thì bọn mình train với nhau 2 contest, chủ yếu là để mọi người trong team phối hợp tốt với nhau thôi chứ mình nghĩ train kiểu này thì trình độ mỗi đứa cũng không có lên nhiều lắm. Ngoài ra tụi mình còn kiểm tra lại nội dung trong notebook để chuẩn bị tốt nhất cho Regional nữa. Trải qua nhiều cuộc thi với nhau nên mình nghĩ bây giờ team mình phối hợp cũng cực kỳ ăn ý với nhau rồi

Hiển: Ở đợt World Finals sắp tới thì cả team có đặt mục tiêu gì không?

Hiệp: Chắc cả team cũng đặt mục tiêu đơn giản là lấy được medal thôi . Ở đợt Regional vừa rồi tổ chức ở trường Sư phạm Kỹ thuật thì bác Long cũng ngồi lại nói chuyện với các team đi World Finals bảo là nếu có team nào đạt được thứ hạng tốt hơn team được medal vừa rồi của Việt Nam thì sẽ được thưởng chục ngàn đô, mình cũng không biết cái đấy thật hay không nhưng mình thấy cái đó cũng khá là vui

Hiển: Vậy để chuẩn bị cho World Finals thì Hiệp sẽ áp dụng chiến thuật cũ hay sẽ áp dụng chiến thuật khác?

Hiệp: Mình nghĩ là cũng phải đổi chiến thuật tại ở World Finals thì không có nhiều bài cho điểm như Regional Việt Nam. Thật ra hiện tại thì team mình cũng chưa làm đề World Finals nào nên cũng chưa có chiến thuật cụ thể, nhưng hiện tại thì mình thiên về hướng đó là có thể cố gắng dành First solve ở World Finals trong cỡ nửa tiếng đầu sau đó thì sẽ dành thời gian đọc hết tất cả các đề và cân nhắc xem những bài có thể làm được thì nhảy vào làm. Ở World Finals thì yêu cầu phải code rất là chắc tay và không được dính pen, tại bình thường mấy đội ở vùng huy chương đồng sẽ ngang nhau về số bài và chỉ hơn nhau một tí pen thôi nên là tụi mình không được phép nộp bài sai và thua người ta về pen. Ngoài ra bọn mình cũng phải chịu khó cày để nâng cao trình độ cho kịp World Finals sắp tới, mình cũng có dự định là học kỳ 2 sắp tới mình sẽ học 6 môn để học kỳ 3 mình có thể có nhiều thời gian để tập trung cho CP.

Hiển: Sau đợt World Finals thì Hiệp có tính thi ICPC vào năm sau để đạt đủ 2 lần dự World Finals không?

Hiệp: World Finals sẽ rơi vào tầm tháng 11 và Regional Việt Nam sẽ vào tháng 12 nên là phải cố gắng cho World Finals lần này. Mình thì cũng rất muốn được dự 2 lần World Finals, năm sau mình được biết là Regional sẽ khá là căng vì sẽ có nhiều team nước ngoài sang thi và có cả team của Trần Xuân Bách đi thi nữa

Hiển: Theo mình được biết thì mỗi Regional sẽ có một bộ bài và dạng bài khác nhau, vậy team của Hiệp thường chọn những contest của Regional nào để luyện tập?

Hiệp: Mình thi ở Regional Việt Nam thì mình

[113]World Finals: Cuộc thi ICPC cuối cùng trong một mùa giải bao gồm những đội xuất sắc nhất của các trường đại học trên thế giới vượt qua vòng loại vùng (Regional contest)

sẽ chủ yếu luyện những contest Regional Việt Nam thôi . Thường thì bọn mình sẽ lên xem thử contest nào bọn mình chưa làm thì sẽ lên các Online Judge để virtual contest đấy, thật ra các contest Regional Việt Nam cũng ít nên là sau khi làm hết đống đấy thì bọn mình sẽ lên CF để xem các team khác làm gì thì bọn mình làm theo . Khi mà virtual contest đấy thì bọn mình cũng xem thử xem phong độ bọn mình có hơn được mấy team đấy không nữa, kiểu mấy team của Vũ Hoàng Kiên rồi anh Nghĩa Phạm ấy

Hiển: Trong các contest mà Hiệp đã virtual thì Hiệp thích bộ đề của Regional nào nhất và tại sao?

Hiệp: Câu hỏi này khó thế, bình thường mình làm xong cũng chả nhớ cái đề nó như nào

Hiển: Hiệp có thần tượng nào trong giới Lập trình thi đấu không?

Hiệp: Chắc hỏi về thần tượng thì ai cũng bảo là tourist[114] thôi . Ở Việt Nam thi chắc là Bùi Hồng Đức hoặc là Vũ Hoàng Kiên, thật sự thì mình cũng chưa tiếp xúc nhiều với Bùi Hồng Đức nên cũng chả biết như thế nào nhưng nhìn bên ngoài thì Bùi Hồng Đức kiểu như là một người khá là lạnh lùng nhưng mà lại cực kỳ khủng. Còn Kiên thì mình thấy trái ngược hẳn với teammate mình là Đăng Huy ấy, Huy thì cuộc đời nó chỉ có mỗi CP với cả Manga thôi . Mình nghĩ là mình thần tượng cả hai người như nhau .

Hiển: Nhắc đến tourist thì tourist có thói quen code là thường dùng space chứ không tab, trùng hợp là Hiệp cũng có thói quen như thế, có phải điều này Hiệp học từ tourist không?

Hiệp: Mình cũng chả biết tourist code như nào cả . Mình thích dùng space hơn phím tab do là nó đồng nhất về khoảng cách và áp dụng được trên nhiều máy khác nhau với cả không cần phải setup độ dài tab khi code, nó chỉ đơn giản vậy thôi

Hiển: Mình được biết là Hiệp có đăng những code giải bài CSES trên GitHub, có phải Hiệp đăng lên như thế để cho mọi người cùng học hỏi không?

Hiệp: Đợt đấy mình đăng là vào Tết 2021, lúc đó mình khá là rảnh và cũng vào đại học rồi ý, chả có gì làm nên là mình làm CSES thử . Một phần nữa là sau thi Học sinh giỏi Quốc gia mình không còn code nên cảm thấy trình độ mình tụt nhiều, thế là mình quyết định đi cày hết các bài CSES, thật ra cũng chưa làm đến mức làm hết tất cả các bài đâu tại mình cũng lười làm tiếp lắm, cơ mà lúc cày xong thì mình cảm giác mình code trâu lên hẳn

Hiển: Hiệp xuất thân là học sinh chuyên Toán, vậy Hiệp cảm thấy lúc học Lập trình thi đấu thì có giúp ích nhiều trong việc giải bài và tư duy khác với những người không có gốc là chuyên Toán không?

Hiệp: Cái đấy mình không nói được vì mình cũng không biết tư duy của các bạn không học chuyên Toán nó như nào . Thật ra mình thấy chắc cũng có giúp ích đấy tại do lúc đấy mình không học về Tin từ trước mà lại lên trình và đạt giải Nhì Quốc gia nhanh như vậy.

Lê Bảo Hiệp tại Văn Miếu - Quốc Tử Giám

Hiển: Bản thân Hiệp là sinh viên năm 3 cũng sắp tốt nghiệp, vậy Hiệp có dự định học Cao học hay đi làm tại các công ty công nghệ không hay sẽ khởi nghiệp?

Hiệp: Từ lúc trước vào đại học thì mình thấy AI đã khá là hot nên mình cũng có dự định đi học PhD để nghiên cứu về AI, nhưng sau khi học xong hai môn liên quan tới AI ở trường thì mình quyết định dẹp luôn . Mình nghĩ rằng cùng lắm mình sẽ phấn đấu học Master và đi làm ở Mỹ chứ mình nghĩ mình không sống nổi 6 năm học PhD

Hiển: Nếu học Master thì Hiệp dự định lĩnh vực học sẽ là gì?

Hiệp: Mình cũng chưa biết đấy sẽ là ngành gì nữa . Mình có kế hoạch là học Master xong thì sẽ ở lại Mỹ xem tình hình thế nào rồi về Singapore sống, tại mình cũng có một đợt du lịch ở Singapore rồi nên thấy sống ở đấy khá là sướng .

Hiển: Thời gian rảnh ngoài cày CP ra thì Hiệp còn chơi game không?

Hiệp: Mình không phải là đứa nghiện game nên mình cũng chả biết nữa, chắc game duy nhất mình chơi là Tetris .

Hiển: Vậy là bạn có sở thích chơi Tetris trong lúc rảnh khá giống bạn Trần Xuân Bách đúng không?

[114]tourist: tên thật là Gennady Korotkevich - một lập trình viên người Belarus. Anh có 6 Huy chương vàng IOI (2007 -– 2012) và 2 lần vô địch thế giới tại kỳ thi ICPC (năm 2013 và 2015).

Hiệp: Không, thật ra là Tetris mình chơi khác với Tetris mà Trần Xuân Bách chơi, nó đau khổ hơn nhiều

Hiển: Đến cuối phần phỏng vấn, Hiệp có lời nào muốn nhắn nhủ đến các bạn độc giả của tạp chí VNOI không? Đặc biệt là các bạn học sinh sắp sửa thi Học sinh giỏi Quốc gia?

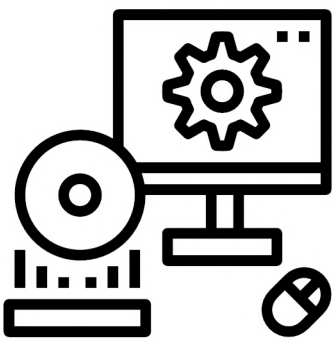
Hiệp: Những gì mình có thể chia sẻ được thì mình cũng đã chia sẻ rồi nên mình cũng không biết nhắn nhủ điều gì đến cho các bạn . Mình chỉ mong muốn rằng các bạn sẽ cố gắng hết sức mình và cho dù có tạch thì cũng đừng nên buồn quá, bởi vì nếu các bạn đã được tham gia kỳ thi này thì chứng tỏ trình độ của bạn đã hơn được rất nhiều người rồi. Việc các bạn được thi môn Tin và học đội tuyển để ôn thi cũng đã cho bạn rất nhiều lợi thế để bước chân vào đại học so với những bạn chưa tiếp xúc với môn Tin bao giờ, nên là dù kết quả không tốt thì các bạn cũng đừng nên thất vọng quá, vì cánh cửa này đóng lại sẽ có cánh cửa khác mở ra cho các bạn.

Hiển: Cảm ơn Hiệp vì một buổi phỏng vấn rất thú vị! Chúc Hiệp luôn nhiều sức khoẻ và thành công trong công việc và sớm thực hiện được ước mơ Gold Medal WF cho Việt Nam!

Hiệp: Cảm ơn bạn nhiều

Shortest Path DAG và ứng dụng

Nguyễn Đăng Quân
Sinh viên năm 1, Trường Đại học Công nghệ, Đại học Quốc gia Hà Nội



Trong một số bài toán liên quan đến đường đi ngắn nhất, tính chất của đồ thị khá khó nhận ra và cũng không dễ dàng để áp dụng. Hôm nay mình xin chia sẻ các bạn một kĩ thuật sử dụng phương pháp tính toán trên DAG (Directed Acyclic Graph) để giải một số bài toán về đường đi ngắn nhất.

Bài toán mở đầu

Đếm số đường đi trên DAG

Cho đồ thị có hướng không chu trình gồm n đỉnh và m cung. Với mỗi đỉnh, hãy đếm số đường đi từ 1 tới đỉnh đó trên đồ thị.

Thuật toán

Bài toán trên khá quen thuộc với đa số mọi người, cách giải là sử dụng qui hoạch động trên Thứ tự topo [115] :

- Đầu tiên, các bạn sắp xếp các đỉnh theo thứ tự Topo: $v_1, v_2, \ldots, v_n$.
- Gọi $f(i)$ là số lượng đường đi từ 1 tới i trên đồ thị, ta có:
  - $f(1) = 1$.
  - $f(v) = \sum f(u)$ với mọi cung $u \to v$ trên đồ thị.

Code mẫu:

```
#include <algorithm>
#include <iostream>
#include <vector>

using namespace std;
const int N = 1e5 + 5;

// adj[u]: Lưu các đỉnh v mà có cung u->v, nếu có
// nhiều cung u->v thì lưu đỉnh v nhiều lần

int n, m;
vector<int> adj[N];
int nTopo, v[N];
bool vis[N];
int f[N];

void dfs(int u) {
  vis[u] = true;

  for (auto i : adj[u])
    if (!vis[i]) dfs(i);

  v[++nTopo] = u;
}

int main() {
  cin >> n >> m;
  for (int i = 1; i <= m; ++i) {
    int u, v;
    cin >> u >> v;
    adj[u].emplace_back(v);
  }

  /* Sắp xếp topo */
  for (int i = 1; i <= n; ++i)
    if (!vis[i]) dfs(i);

  // Sau khi dfs xong, ta được thứ tự trong v là
  // thứ tự ngược topo
  reverse(v + 1, v + nTopo + 1);

  /* Qui hoạch động trên DAG */
  for (int i = 1; i <= n; ++i)
    for (auto j : adj[v[i]]) f[j] += f[v[i]];

  // In đáp án
  for (int i = 1; i <= n; ++i)
    cout << f[i] << " ";
}
```

Khi đó, độ phức tạp của thuật toán là $O(m + n)$, bởi thao tác sắp xếp Topo và thao tác tính hàm mục tiêu đều tốn thời gian $O(m + n)$.

Bài toán đếm số đường đi ngắn nhất

Cho đồ thị vô hướng không chu trình $S = (V, E)$ gồm n đỉnh và m cạnh có trọng số **dương**. Với mỗi đỉnh, hãy đếm số đường đi ngắn nhất từ 1 tới đỉnh đó trên đồ thị.

Link bài tập gốc
https://vjudge.net/problem/Gym-406204J

Hướng tiếp cận

Trước tiên; để đếm được số đường đi ngắn nhất, ta cần phải nắm được cách tìm giá trị đường đi đó. Mình xin phép không nhắc lại các thuật toán tìm kiếm đường đi ngắn nhất; nếu bạn chưa biết hoặc biết nhưng đã quên thì có thể đọc lại trên VNOI wiki [116] .

Trong bài toán này, ta sẽ gọi:

- $d(i)$ là độ dài đường đi ngắn nhất từ đỉnh 1 tới đỉnh i trên đồ thị.
- $f(i)$ là số đường đi ngắn nhất từ đỉnh 1 tới đỉnh i.

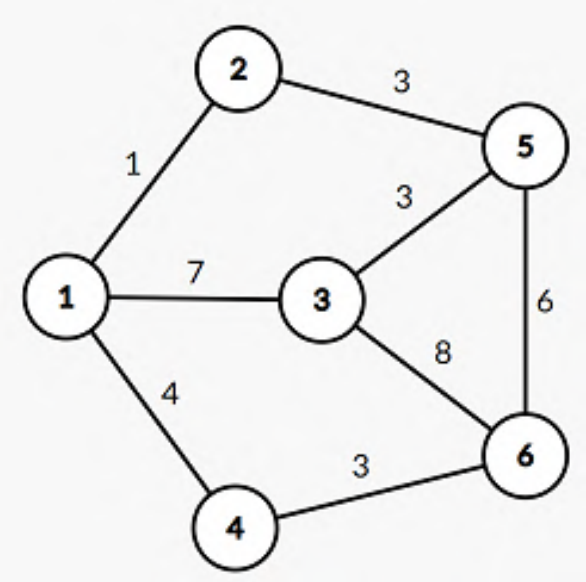
Vậy tại sao phải tính được trước độ dài đường đi ngắn nhất?

Định hướng lại đồ thị S như sau:

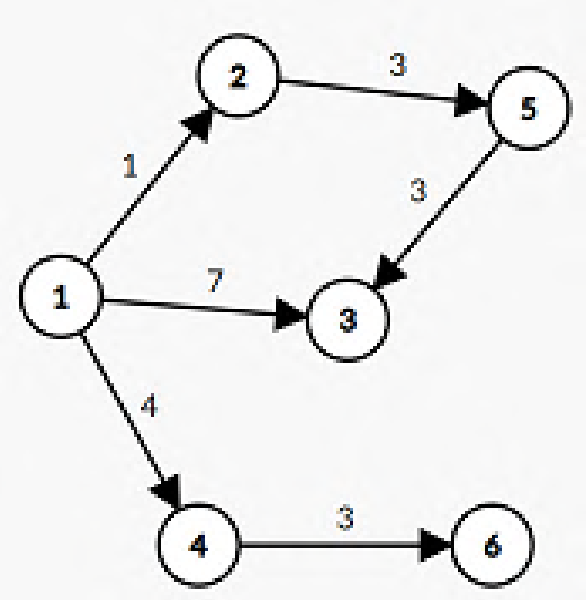
- Giữ nguyên các đỉnh
- Với mỗi cạnh $u - v \in E$
  - Nếu $d(u) + w(u, v) = d(v)$, cạnh này sẽ biến thành cung $u \to v$.
  - Nếu $d(v) + w(u, v) = d(u)$, cạnh này sẽ biến thành cung $v \to v$.
  - Nếu hai điều kiện trên không thỏa mãn, cạnh này bị xóa bỏ khỏi đồ thị.

[115] https://vnoi.info/wiki/algo/graph-theory/topological-sort.md
[116] https://vnoi.info/wiki/algo/graph-theory/shortest-path.md

Ví dụ:



Đồ thị ban đầu



Đồ thị đã thay đổi

Khi đó, $f(i)$ chính là số đường đi từ 1 đến i. Bài toán này trở thành bài *Đếm số đường đi trên DAG*.

Để hiểu rõ hơn, các bạn hãy theo dõi phần tiếp theo.

Shortest Path DAG

Cho một đồ thị vô hướng $S = (V, E)$. Gọi $d(i, j)$ là **độ dài** đường đi ngắn nhất từ i đến j (Nếu không tồn tại đường đi thì $d(i, j) = \infty$).

Ta dựng ra một đồ thị *có hướng* $S'_u = (V, E'_u)$ như sau:

- Mỗi đỉnh trong đồ thị S sẽ ứng với một đỉnh trong S'_u.
- Mỗi cạnh $i - j \in E$ sẽ biến đổi như sau:
  - Nếu $d(u, i) + w(i, j) = d(u, j)$ thì dựng ra cung $i \to j$ trong E'_u.
  - Nếu $d(u, j) + w(i, j) = d(u, i)$ thì dựng ra cung $j \to i$ trong E'_u.

Khi đó, ta gọi đồ thị S'_u là Shortest Path DAG ứng với đỉnh u trên S.

Tính chất

Đồ thị S'_u có một số tính chất khá đặc biệt như sau:

1. Nếu tồn tại đường đi từ u đến v trên S (Tức là $d(u, v) < \infty$) thì cũng tồn tại đường đi (có hướng) từ u đến v trên S'_u

Đây là một nhận xét hiển nhiên, bạn đọc có thể tự suy ngẫm về tính đúng.

1. Mọi đường đi trên S'_u đều là đường đi ngắn nhất trên S.

Chứng minh: Gọi $f(i, j)$ là độ dài đường đi **dài nhất** từ i đến j trên S'_u (Nếu không tồn tại thì $f(i, j) = \infty$). Điều kiện trên tương đương với một trong hai điều kiện sau thỏa mãn:

- $f(i, j) = \infty$ (Không tồn tại đường đi)
- $f(i, j) = d(i, j)$ (Nếu tồn tại đường đi thì đó phải là đường đi ngắn nhất)

Trước tiên, ta chứng minh đường đi xuất phát từ đỉnh u đều là đường đi ngắn nhất. Giả sử nhận xét trên là sai, tức là tồn tại một đỉnh v và đường đi

$$u \to x_1 \to x_2 \to \cdots \to x_k$$

mà $w(u, x_1) + w(x_1, x_2) + \cdots + w(x_k, v) > d(u, v)$.

Theo đó:

$$\begin{aligned} & [d(u, x_1) - d(u, u)] + [d(u, x_2) - d(u, x_1)] + \\ & \quad + \cdots + d(u, v) - d(u, x_k)] > d(u, v) \\ \Leftrightarrow \quad & d(u, v) - d(u, u) > d(u, v) \\ \Leftrightarrow \quad & 0 > 0 \end{aligned}$$

$\Rightarrow$ Điều ta giả sử là sai, dẫn đến $f(u, i) = d(u, i) \forall i$

Nếu tồn tại đường đi từ i đến j, ta biết $d(u, i) < \infty$; do đó tồn tại đường đi từ u đến i và $d(u, i) + f(i, j) = d(u, j)$.

Mặt khác, theo bất đẳng thức tam giác [117] :

$$d(u, i) + d(i, j) \geq d(u, j)$$

[117] https://sharmaeklavya2.github.io/theoremdep/nodes/graph-theory/shortest-paths/triangle-inequality.html

hay

$$d(u,i) + d(i,j) \geq d(u,i) + f(i,j)$$

suy ra

$$d(i,j) \geq f(i,j)$$

Dấu bằng phải xảy ra, hay $f(i,j) = d(i,j)$. Ta hoàn tất chứng minh.

1. S' là đồ thị có hướng **không chu trình** (DAG)

Chứng minh: Nếu đồ thị có chu trình chứa một đỉnh v nào đó, ta có $d(v,v) > 0$ do trọng số các cạnh là dương, dẫn đến mâu thuẫn.

1. Nếu tồn tại đường đi từ i đến j trên S'_u thì **mọi đường đi ngắn nhất** từ i đến j sẽ đều xuất hiện trong S'_u.

Chứng minh: Vì tồn tại một đường đi từ i đến j trên S'_u nên theo tính chất 2: $d(u,i) + d(i,j) = d(u,i) + f(i,j) = d(u,j)$

Xét một **đường đi ngắn nhất** từ i đến j không xuất hiện trong đồ thị:

$$i \to x_1 \to x_2 \to \cdots \to x_k \to j$$

Theo bất đẳng thức tam giác:

$$\begin{cases} d(u,x_1) - d(u,i) \leq d(i,x_1) \\ d(u,x_2) - d(u,x_1) \leq d(x_1,x_2) \\ \cdots \\ d(u,j) - d(u,x_k) \leq d(x_k,j) \end{cases}$$

Cộng về với vế ta được:

$$d(u,j) - d(u,i) \leq d(i,x_1) + d(x_1,x_2) + \cdots + d(x_k,j)$$
$$\Leftrightarrow d(i,j) \leq d(i,x_1) + d(x_1,x_2) + \cdots + d(x_k,j)$$

Mặt khác ta có đường đi trên là đường đi ngắn nhất nên $d(i,j) = w(i,x_1) + w(x_1,x_2) + \cdots + w(x_k,j)$ Theo đó:

$$w(i,x_1) + w(x_1,x_2) + \cdots + w(x_k,j) \leq d(i,x_1) + d(x_1,x_2) + \cdots + d(x_k,j)$$

Dấu bằng phải xảy ra, dẫn đến dấu bằng của bất đẳng thức tam giác phải xảy ra:

$$\begin{cases} d(u,i) + d(i,x_1) = d(u,x_1) \\ d(u,x_1) + d(x_1,x_2) = d(u,x_2) \\ \cdots \\ d(u,x_k) + d(x_k,j) = d(u,j) \end{cases}$$

Theo đó, đường đi trên phải nằm trong S'_u. Tính chất số 4 chính là chìa khóa giúp ta giải quyết được *Bài toán đếm số đường đi ngắn nhất_.

Cài đặt

Shortest Path DAG có ý nghĩa về mặt đưa ra nhận xét, các tính chất hơn thay vì được coi như một cấu trúc dữ liệu.

Vì vậy, thông thường mình sẽ không dựng lại cả một đồ thị mới mà sẽ tận dụng lại tài nguyên (Danh sách đỉnh, danh sách cạnh,...) của đồ thị cũ. Chỉ cần chú ý khi duyệt đồ thị, ta cần kiểm tra tính tồn tại của cung dựa trên điều kiện của DAG.

Áp dụng

Sử dụng các tính chất trên, ta có thể giải quyết một số bài toán về đường đi ngắn nhất; sẽ khá khó để có thể giải quyết các bài toán sau đây với những công cụ cơ bản.

VNUOI 2022: DELETE

Cho một đồ thị vô hướng gồm n đỉnh và $m+k$ cạnh với trọng số duonwg; trong đó có m cạnh loại 1 và k cạnh loại 2. Các cạnh loại 2 chỉ nối giữa đỉnh 1 và một đỉnh khác.

Yêu cầu: Hãy đếm số cách xóa một số cạnh loại 2 (Có thể không xóa cạnh nào) sao cho đường đi ngắn nhất từ đỉnh 1 tới các đỉnh khác là **không đổi**.

Bài toán có thể xem tại
https://oj.vnoi.info/problem/vnuoi22_delete

Nhận xét Trong bài toán trên, điều kiện của các cạnh loại 2 sẽ giúp ta giải bài toán dễ hơn. Tuy nhiên, nếu không có điều kiện ấy thì bài toán vẫn giải được dựa vào *Shortest Path DAG*.

Xét Shortest Path DAG S'_1 của đồ thị, S'_1 sẽ thay đổi ứng với mỗi thay đổi (thêm/xóa cạnh) trên đồ thị ban đầu; ta có một số nhận xét:

- Nếu một cạnh không tạo ra một cung trong S'_1, việc xóa nó không ảnh hưởng đến dường đi ngắn nhất từ 1 tới các đỉnh. Lượng đóng góp của cạnh này là 2.
- Nếu vẫn tồn tại đường đi từ đỉnh 1 đến đỉnh i trên S'_1 thì độ dài đường đi ngắn nhất từ 1 đến i trên đồ thị ban đầu là không đổi. (Tính chất số 1 và Tính chất số 2).

$\Rightarrow$ Ta qui việc đếm cách xóa cạnh trên đồ thị ban đầu về đếm số cách xóa cung trên S'_1.

Mặt khác, do S'_1 là DAG (Tính chất số 3) $\Rightarrow$ Sau khi xóa cung, các đỉnh ban đầu có bậc vào khác 0 thì sau khi xóa cũng phải có bậc vào khác

0. ⇒ Ta đếm số cách xóa để các đỉnh có bậc vào khác 0 vẫn tiếp tục khác 0.

Thuật toán Trước tiên, ta sẽ dựng đồ thị S_1' của đồ thị đã cho. Xét đỉnh i, gọi:

- $deg(i)$ là bậc vào của đỉnh i
- $deg_1(i)$ là số cung "loại 1" vào đỉnh i
- $deg_2(i)$ là số cung "loại 2" vào đỉnh i

Lượng đóng góp của đỉnh i, tức là số cách xóa cung vào để bậc của nó tiếp tục bằng 0 hoặc tiếp tục khác 0 là:

- 1 nếu $deg(i) = 0$
- $2^{deg_2(i)}$ nếu $deg_1(i) > 0$ vì $deg(i) = deg_1(i) + deg_2(i) \geq deg_1(i) > 0$
- $2^{deg_2(i)} - 1$ nếu $deg_1(i) = 0$ vì $deg(i) = deg_2(i)$

Từ đó, đáp án là **tích** lượng đóng góp của các đỉnh và các cạnh loại 2 sinh ra cung trong S_1'.

Code mẫu:

```
#include <cstdio>
#include <iostream>
#include <queue>
#include <vector>

using namespace std;
using ll = long long;

constexpr int N = 2e5 + 5;
constexpr int M = 5e5;
constexpr ll mod = 998244353;
constexpr ll Inf = 1e17;

struct Edge {
  int u, v, w;
} s1[M], s2[N];

int n, k, m;
int deg_1[N], deg_2[N];
ll d[N];
// adj[i] = đỉnh kề thông qua các cạnh loại 1
// nadj[i] = đỉnh kề thông qua các cạnh loại 2
vector<pair<int, int>> adj[N], nadj[N];

// Nhập dữ liệu
void Read() {
  cin >> n >> m >> k;
  for (int i = 1; i <= m; ++i) {
    cin >> s1[i].u >> s1[i].v >> s1[i].w;
    adj[s1[i].u].emplace_back(s1[i].v, i);
    adj[s1[i].v].emplace_back(s1[i].u, i);
  }

  for (int i = 1; i <= k; ++i) {
    s2[i].u = 1;
    cin >> s2[i].v >> s2[i].w;
    nadj[s2[i].u].emplace_back(s2[i].v, i);
    nadj[s2[i].v].emplace_back(s2[i].u, i);
  }
}

// Tính a ^ b % mod
ll Pow(ll a, ll b) {
  ll ans(1);

  for (; b; b >>= 1) {
    if (b & 1) ans = ans * a % mod;
    a = a * a % mod;
  }
  return ans;
}

// Thuật toán tìm đường đi ngắn nhất
// Kết quả trả về mảng d[], d[i] = đường đi ngắn
// nhất từ x đến i
void ShortestPath(int x) {
  // Bạn đọc vui lòng tự cài đặt lại thuật toán
}

// Thực hiện thuật toán
void Solve() {
  ll ans(1);
  ShortestPath(1);

  /* Dựng Shortest Path DAG */

  for (int i = 1; i <= m; ++i)
    if (d[s1[i].u] + s1[i].w == d[s1[i].v])
      ++deg_1[s1[i].v];
    else if (d[s1[i].v] + s1[i].w == d[s1[i].u])
      ++deg_1[s1[i].u] = 1;

  for (int i = 1; i <= k; ++i)
    if (d[s2[i].u] + s2[i].w == d[s2[i].v])
      ++deg_2[s2[i].v];
    else if (d[s2[i].v] + s2[i].w == d[s2[i].u])
      ++deg_2[s2[i].u];
    else
      ans =
        ans * 2 % mod; // Cạnh này không sinh cung

  /* Kết thúc dựng Shortest Path DAG */

  for (int i = 2; i <= n; ++i)
    if (deg_1[i] > 0)
      ans = ans * Pow(2, deg_2[i]) % mod;
    else if (deg_2[i] > 0)
      ans = ans * (Pow(2, deg_2[i]) - 1) % mod;

  cout << (ans + mod) % mod;
}

int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  cout.tie(0);
  Read();
  Solve();
}
```

Nhìn đoạn code trên, ta có thể dễ thấy độ phức tạp thời gian của thuật toán là $O(m + nlog_2n + ShortestPath)$. Thuật toán chạy nhanh hay chậm phụ thuộc phần nhiều vào cách cài đặt hàm ShortestPath.

BIẾN ĐỘNG

Cho một đồ thị vô hướng **liên thông** gồm n đỉnh và m cạnh có trọng số dương. Gọi $d(u, v)$ là độ dài đường đi ngắn nhất giữa u và v. **Yêu cầu:** Với mỗi cạnh, hãy đếm số cặp đỉnh (u, v) mà $d(u, v)$ tăng nếu ta xóa bỏ cạnh này.

Bài toán có thể xem tại https://vjudge.net/problem/Gym-406204L

Nhận xét Bài toán yêu cầu với mỗi cạnh, đếm số cặp đỉnh (u, v) mà $d(u, v)$ tăng lên khi xóa cạnh

đó. Vậy cạnh này có điều kiện gì mà ảnh hưởng được đến đường đi ngắn nhất từ u đến một đỉnh khác? Rõ ràng cạnh này phải thuộc một đường đi ngắn nhất xuất phát từ đỉnh u!

Ta dựng ra Shortest Path DAG S'_u; gọi $cnt(i, j)$ là số đường đi từ đỉnh i đến đỉnh j. Nếu xóa cung $i \to j$ làm $d(u, v)$ tăng thì mọi đường đi từ u đến v phải thông qua cung này, tức là:

$$cnt(u, i) \times cnt(j, v) = cnt(u, v)_{(*)}$$

Từ nhận xét ở bài *DELETE* ta có thể xóa bỏ các cung sao cho mỗi đỉnh trên S'_u có bậc tối đa là 1 và đường đi ngắn nhất từ u đến các đỉnh là không đổi.

$\Rightarrow$ Các "ứng cử viên" là các cung được giữ lại. Có thể chứng minh số cung của S'_u khi ấy không quá $n - 1$.

Thuật toán Ta dựng ra các Shortest Path DAG $S'_1, S'_2, \ldots, S'_n$. Trên S'_u:

- Ta duyệt qua các "ứng cử viên" mà trong phần nhận xét đã nói
- Với mỗi ứng cử viên, ta lại duyệt các đỉnh v nhằm đếm số lượng đường đi $d(u, v)$ bị tăng nếu xóa "ứng cử viên" đó (Dựa vào điều kiện $(*)$)

Chú ý:

- Theo tính chất 4, $cnt(i, j)$ chính là số đường đi ngắn nhất từ i đến j trên đồ thị ban đầu $\Rightarrow$ Ta có thể tính trước đường đi ngắn nhất giữa mọi cặp đỉnh bằng cách thực hiện n lần thuật toán Dijkstra cổ điển (Thuật toán Dijkstra với độ phức tạp $O(n^2 + m)$)
- $cnt(i, j)$ có thể rất lớn, nên bạn hãy lưu số này trong modulo một số nào đó (Hoặc có thể trong hai modulo); khi so sánh ta sử dụng đồng dư thay vì bằng nhau hoàn toàn.

Vì số lượng ứng cử viên của mỗi DAG là $n - 1$ nên độ phức tạp thời gian khi ấy là $O(n^2 \times (n - 1)) + O(n \times (n^2 + m)) = O(n^3 + nm)$ là chi phí dựng n DAG và chi phí duyệt.

Code mẫu:

```cpp
#include <algorithm>
#include <cstdio>
#include <cstring>
#include <iostream>
#include <vector>

using namespace std;
using ll = long long;

constexpr int N = 5e2 + 5;
constexpr ll Inf = 1e17;
// Mình tính cnt(i,j) trong hai modulo nguyên tố
// là 1e9+7 và 998244353
constexpr int mod[2] = {1000000007, 998244353};

inline void
Add(int &x, const int &y,
    const int
      &mod) { // Chỉ là template cộng mod mà thôi
  x += y;
  if (x >= mod) x -= mod;
}

int n, m;

struct Edge {
  int u, v, w;
} e[N * N];
int ans[N * N];

vector<pair<int, int>> adj[N];
vector<int> candidate;

ll d[N][N];
int cnt[N][N][2];
int mark, check[N];

// Thuật toán Dijkstra cổ điển
void Dijkstra(int x, ll d[N]) {
  ++mark;
  fill_n(d, N, Inf);
  d[x] = 0;
  cnt[x][x][0] = cnt[x][x][1] = 1;

  for (int i = 1; i <= n; ++i) {
    pair<ll, int> ans = {Inf, 0};

    for (int j = 1; j <= n;
         ++j) // Thay vì dùng Hàng đợi ưu tiên, ta
              // duyệt lại toàn bộ
      if (check[j] != mark && d[j] < ans.first)
        ans = {d[j], j};

    if (ans.first == Inf) break;

    check[ans.second] = mark;

    for (auto i : adj[ans.second])
      if (d[i.first] >
          ans.first + e[i.second].w) {
        d[i.first] = ans.first + e[i.second].w;
        cnt[x][i.first][0] =
          cnt[x][ans.second][0];
        cnt[x][i.first][1] =
          cnt[x][ans.second][1];
      } else if (d[i.first] ==
                 ans.first + e[i.second].w) {
        Add(cnt[x][i.first][0],
            cnt[x][ans.second][0], mod[0]);
        Add(cnt[x][i.first][1],
            cnt[x][ans.second][1], mod[1]);
      }
  }
}

// dfs để tìm tập "ứng cử viên"
void dfs(int v, int p, ll d[N]) {

  for (auto i : adj[v])
    if (i.first != p &&
        d[i.first] == d[v] + e[i.second].w) {
      candidate.emplace_back(i.second);
      dfs(i.first, v, d);
    }
}

int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  cout.tie(0);

  cin >> n >> m;
```

```
  for (int i = 1; i <= m; ++i) {
    cin >> e[i].u >> e[i].v >> e[i].w;
    adj[e[i].u].emplace_back(e[i].v, i);
    adj[e[i].v].emplace_back(e[i].u, i);
  }

  // Thực hiện n lần Dijkstra và cho chạy thuật
  // toán đếm đường đi ngắn nhất song song với
  // Dijkstra
  for (int i = 1; i <= n; ++i) Dijkstra(i, d[i]);

  for (int i = 1; i <= n; ++i) {
    candidate.clear();

    dfs(i, -1, d[i]);

    for (auto j : candidate) {
      // Xác định chiều của cung
      int v = d[i][e[j].u] > d[i][e[j].v]
                ? e[j].u
                : e[j].v;
      int u = e[j].u + e[j].v - v;

      for (int t = 1; t <= n; ++t)
        if (d[i][u] + e[j].w + d[v][t] ==
              d[i][t] &&
            1 ll * cnt[i][u][0] * cnt[v][t][0] %
                mod[0] ==
              cnt[i][t][0] &&
            1 ll * cnt[i][u][1] * cnt[v][t][1] %
                mod[1] ==
              cnt[i][t][1])
          ++ans[j];
    }
  }

  for (int i = 1; i <= m; ++i)
    cout
      << ans[i] / 2
      << "\n"; // Vì mỗi cặp đỉnh được tính hai
               // lần nên kết quả phải chia cho 2
}
```

Luyện tập

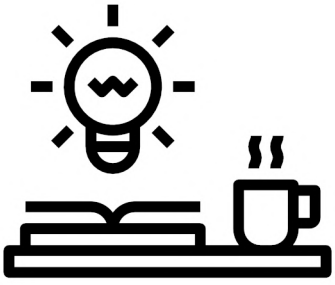
- Codeforces[118] 449B - Jzzhu and Cities [119]
- COCI 2008 - Najkraci [120] (aka PVHOI 2020 - CAULUONG [121])
- Gym - Đường ai nấy đi [122]

[118] Codeforces: một trang web tổ chức các cuộc thi lập trình thi đấu với các dạng đề, bài tập đa dạng.
[119] https://codeforces.com/problemset/problem/449/B
[120] https://dmoj.ca/problem/coci08c3p6
[121] https://1drv.ms/b/s!Asql3vhziCu10CZ5FDu69rbPIDyM?e=6yie4Z
[122] https://vjudge.net/problem/Gym-406204K

Cây DSU

(DSU Tree)

Lê Minh Khánh
12A1 Tin, Trường THPT Chuyên Khoa học Tự nhiên, Đại học Quốc gia Hà Nội



Lời nói đầu

Một số bài viết nên đọc trước khi tiếp tục:

- Disjoint Set Union [123]
- Bài toán RMQ và bài toán LCA [124]
- Thuật toán Kruskal [125]

Cây DSU là một kỹ thuật không phải là quá phức tạp hay xuất hiện quá ít trong giới Competitive Programming, nhưng kỳ lạ thay những bài viết về nó lại chỉ đếm được bằng đầu ngón tay. Cái tên "Cây DSU" không phải là một cái tên chính thức của kỹ thuật này, nó còn xuất hiện dưới nhiều cái tên khác như Reachability Tree hay Kruskal Reconstruction Tree. Không ai rõ kỹ thuật này xuất hiện từ lúc nào, cá nhân mình cũng chỉ biết được rằng những lời giải sử dụng kỹ thuật này đã xuất hiện vào những năm 2000 ở Trung Quốc.

Vậy cây DSU được sử dụng để làm gì? Sức mạnh của cây DSU nằm ở việc nó có thể biểu diễn các thành phần liên thông dưới dạng cây. Từ đây, chúng ta có thể sử dụng được Euler Tour cùng với các cấu trúc dữ liệu xử lý đoạn.

Cách xây dựng từ một đồ thị gốc

Cây DSU là một cây có gốc gồm $N + M$ đỉnh với N và M lần lượt là số đỉnh và số cạnh của đồ thị gốc. Trong đó, cây DSU sẽ có N lá biểu diễn cho các đỉnh ở trong đồ thị gốc, M đỉnh còn lại sẽ biểu diễn cho các cạnh.

Để xây dựng cây DSU, chúng ta khởi tạo N đỉnh đầu tiên, rồi duyệt qua từng cạnh một theo độ lớn tăng dần (giảm dần). Khi thêm cạnh (u, v), chúng ta tạo thêm một đỉnh ảo để biểu diễn cho cạnh này đồng thời gán nó làm cha của đại diện thành phần liên thông của lần lượt u và v. Trong một số bài toán, chúng ta có thể bỏ qua những cạnh nối hai đỉnh cùng một thành phần liên thông và từ đó có được cây DSU gồm $2N - 1$ đỉnh.

Cài đặt

Sau khi hiểu được cây DSU được cấu tạo nên như thế nào, xây dựng cây DSU là một công việc khá dễ dàng và nhanh gọn khi chúng ta chỉ cần chỉnh sửa hàm $find$ và $join$ của DSU một chút như sau:

```
// lưu thông tin cha của các đỉnh trong cây DSU
int par[MAXN];
// vector lưu đồ thị cây DSU
vector<int> g[MAXN];
int n; // lưu số đỉnh của cây

// cài đặt tương tự như DSU thông thường
int find(int u) {
  return (par[u] == u) ? u
                       : par[u] = find(par[u]);
}

void join(int u, int v) {
  u = find(u);
  v = find(v);
  // tạo thêm một đỉnh đại diện cho cạnh
  n++;
  par[n] = n;
  par[u] = n;
  par[v] = n;
  g[n].push_back(u);
  g[n].push_back(v);
}
```

Độ phức tạp thời gian

Có 2 điều chúng ta cần chú ý khi xây dựng cây DSU:

- Không thể sử dụng phương pháp tối ưu gộp theo kích cỡ / độ cao như trong DSU thông thường
- Vẫn có thể sử dụng phương pháp tối ưu nén đường đi như trong DSU

Như đã được phân tích chi tiết ở trong bài viết khác [126] về DSU ở trên VNOI Wiki, nếu chúng ta chỉ sử dụng phương pháp tối ưu nén đường đi thì độ phức tạp trung bình của thao tác $find$ cũng chỉ là $O(log(n))$. Do đó độ phức tạp thời gian của việc xây dựng cây DSU chỉ là $O(n * log(n))$.

Bài toán 1: Path Max Queries

Đề bài

Cho một cây gồm N đỉnh và $N - 1$ cạnh có trọng số. Với mỗi truy vấn, hãy tìm cạnh có trọng số lớn nhất trên đường đi từ đỉnh u đến đỉnh v.

Nhận xét

Không quá khó để chúng ta giải quyết bài toán trên bằng cách sử dụng kỹ thuật Heavy-Light Decomposition hay Binary Lifting. Nhưng với hai giải

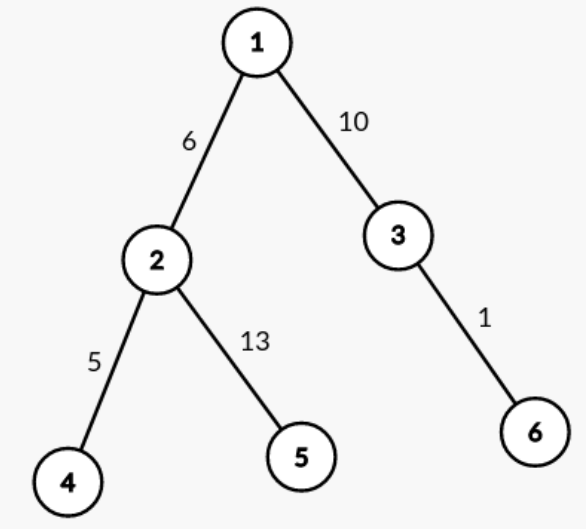
[123] https://vnoi.info/wiki/algo/data-structures/disjoint-set-union.md
[124] https://vnoi.info/wiki/translate/topcoder/Range-Minimum-Query-and-Lowest-Common-Ancestor.md
[125] https://vnoi.info/wiki/algo/graph-theory/minimum-spanning-tree.md#1-thu%E1%BA%ADt-to%C3%A1n-kruskal
[126] https://vnoi.info/wiki/algo/data-structures/disjoint-set-union.md

pháp ở trên, độ phức tạp cho mỗi truy vấn lần lượt là $O(log^2(n))$ và $O(log(n))$. Liệu có cách nào để chúng ta tiền xử lý cây trong $O(n * log(n))$ và trả lời mỗi truy vấn với độ phức tạp chỉ là $O(1)$ hay không?

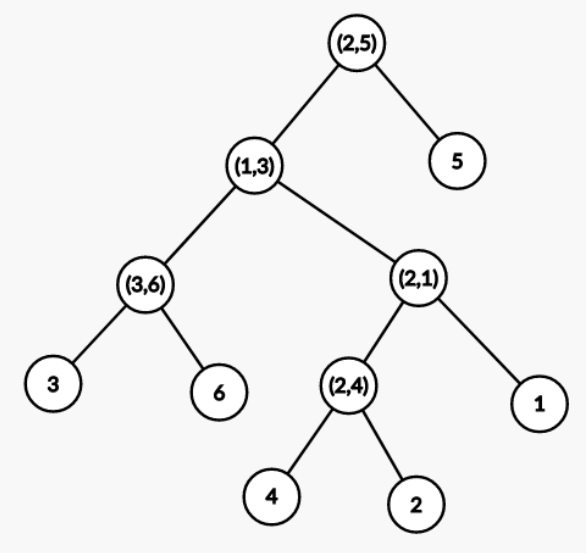
Câu trả lời là có! Và nó nằm ở chính cây DSU.

Lời giải

Chúng ta sẽ sort các cạnh theo trọng số tăng dần rồi sau đó xây dựng lên cây DSU (tương tự như thuật toán Kruskal). Nhìn vào hai đồ thị bên dưới chúng ta có thể đưa ra một nhận xét: cạnh có trọng số lớn nhất trên đường đi từ đỉnh u đến v chính là LCA của chúng trên cây DSU.



Hình ảnh của đồ thị gốc



Hình ảnh của cây DSU đã được xây dựng

Mặt khác, từ bài viết về RMQ và LCA [127] , chúng ta đã biết rằng việc tính toán LCA trong $O(1)$ bằng cách sử dụng Euler Tour và cấu trúc dữ liệu chẳng hạn như Sparse Table là điều hoàn toàn có thể.

Như vậy, bài toán đã được giải quyết! Độ phức tạp: $O(N * log(N) + Q)$ với N và Q lần lượt là số đỉnh và số truy vấn.

Bài toán 2: Codeforces 1416D - Graph and queries

Tóm tắt

Cho đồ thị N đỉnh và M cạnh. Mỗi đỉnh có một số hiệu p_i đôi một khác nhau được viết lên nó. Xử lý các loại truy vấn sau

- Truy vấn loại 1: $1\ v$ - Tìm đỉnh u cùng thành phần liên thông với đỉnh v có số hiệu lớn nhất. In ra số hiệu được viết lên đỉnh u và thay đổi số hiệu của u thành 0.
- Truy vấn loại 2: $2\ i$ - Xóa cạnh thứ i của đồ thị.

Lời giải

Ở các bài toán có xuất hiện truy vấn xóa cạnh, do tính một chiều của cấu trúc dữ liệu DSU, chúng ta có thể nghĩ đến cách đảo ngược thứ tự của các truy vấn của bài để giải.

Chúng ta có thể xử lý bài này như sau:

- Thực hiện nối các cạnh không bị xóa
- Thêm dần các cạnh bị xóa theo thứ tự từ dưới lên bằng cách sử dụng DSU
- Áp dụng cây DSU từ đó xác định được thành phần liên thông của đỉnh v tại mọi thời điểm
- Sử dụng Euler Tour và Segment Tree để xử lý bài toán tìm max trong khoảng và update tại một điểm

Độ phức tạp: $O((n + m + q) * log(n))$

Giá trị của cây DSU trong bài tập này chính là biểu diễn các thành phần liên thông tại từng thời điểm dưới dạng cây, thứ mà chúng ta quen thuộc hơn và có nhiều công cụ để giải quyết các bài toán hơn.

Cài đặt

```
#include <bits/stdc++.h>

using namespace std;
int const MAXN = 7e5 + 5;
int n, m, q;
int label[MAXN];
pair<int, int> edge[MAXN], query[MAXN];
```

[127] https://vnoi.info/wiki/translate/topcoder/Range-Minimum-Query-and-Lowest-Common-Ancestor

```
bool del[MAXN];

// Xây dựng cây DSU
vector<int> g[MAXN];
int par[MAXN];

int find(int u) {
  if (par[u] == u) return u;
  return par[u] = find(par[u]);
}

void join(int u, int v) {
  u = find(u);
  v = find(v);
  if (u == v) return;
  n++;
  par[n] = n;
  par[u] = n;
  par[v] = n;
  g[n].push_back(u);
  g[n].push_back(v);
}

// Xây dựng Euler Tour trên cây DSU
int tin[MAXN], tout[MAXN];
int now = 0;
void dfs_time(int u, int p = -1) {
  tin[u] = ++now;
  for (int v : g[u]) {
    if (v == p) continue;
    dfs_time(v, u);
  }
  tout[u] = now;
}

// Segment tree trên Euler Tour để xử lý truy vấn
// 1
#define left node << 1, tl, tm
#define right node << 1 | 1, tm + 1, tr
pair<int, int> st[MAXN << 2];

void update(pair<int, int> val, int pos,
            int node = 1, int tl = 1,
            int tr = now) {
  if (tl == tr) {
    st[node] = val;
    return;
  }
  int tm = (tl + tr) >> 1;
  if (pos <= tm) update(val, pos, left);
  else
    update(val, pos, right);
  st[node] =
    max(st[node << 1], st[node << 1 | 1]);
}

pair<int, int> get(int l, int r, int node = 1,
                   int tl = 1, int tr = now) {
  if (l > r || r < tl || l > tr)
    return make_pair(0, 0);
  if (l <= tl && tr <= r) { return st[node]; }
  int tm = (tl + tr) >> 1;
  return max(get(l, r, left), get(l, r, right));
}

void solve() {
  cin >> n >> m >> q;
  for (int i = 1; i <= n; i++) {
    cin >> label[i];
  }
  for (int i = 1; i <= m; i++) {
    cin >> edge[i].first >> edge[i].second;
  }
  for (int i = 1; i <= q; i++) {
    cin >> query[i].first >> query[i].second;
    if (query[i].first == 2) {
      del[query[i].second] = true;
    }
  }
  // setup dsu căn bản
  for (int i = 1; i <= n; i++) { par[i] = i; }
  // nối các cạnh không bị xóa
  for (int i = 1; i <= m; i++) {
    if (del[i]) continue;
    join(edge[i].first, edge[i].second);
  }

  // xác định thành phần liên thông của đỉnh v mỗi
  // khi bị truy vấn
  for (int i = q; i >= 1; i--) {
    if (query[i].first == 2) {
      int id = query[i].second;
      join(edge[id].first, edge[id].second);
    } else {
      query[i].second = find(query[i].second);
    }
  }

  // xây dựng euler tour trên dsu-tree
  for (int i = 1; i <= n; i++) {
    if (par[i] != i) continue;
    dfs_time(i);
  }
  // đặt các số hiệu về giá trị gốc vào segment
  // tree trên euler tour
  for (int i = 1; i <= n; i++) {
    update({label[i], tin[i]}, tin[i]);
  }
  // xử lý truy vấn
  for (int i = 1; i <= q; i++) {
    if (query[i].first == 2) continue;
    int root =
      query[i]
        .second; // tìm thành phần liên thông của
                 // đỉnh v trong cây DSU vào thời
                 // điểm hiện tại
    pair<int, int> ans = get(
      tin[root], tout[root]); // truy vấn tìm max
                              // trên Segment Tree
    cout << ans.first << endl;
    if (ans.first != 0) {
      update(
        {0, ans.second},
        ans.second); // cập nhật số hiệu đỉnh u
                     // thành 0 trên Segment Tree
    }
  }
}

signed main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  cout.tie(0);
  solve();
}
```

Hoặc các bạn có thể theo dõi tại submission trên Codeforces [128] .

Tổng kết

Ở trên là tất cả những gì căn bản nhất của cây DSU. Mong rằng qua bài viết này các quý độc giả đã có được thêm một công cụ hữu ích để giải quyết các bài tập trong tương lai.

[128] https://codeforces.com/contest/1416/submission/188630658

Một số bài tập để luyện tập kỹ thuật

- Werewolf IOI[129] 2018 [130]
- USACO 2014 January Contest Gold - Ski Course Rating [131]
- Codechef TULIPS - Tiptoe through the tulips [132]
- Codechef CHEFCOMP - Chefina and Strange Tree [133]
- APIO[134] 2020 - Swapping Cities [135]
- Codeforces 1706E - Qpwoeirut and Vertices [136]
- Codeforces 1628E Groceries in Meteor Town [137]

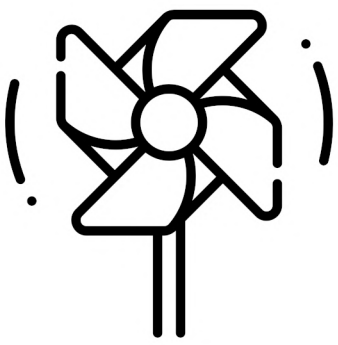
Tham khảo

- Reachability Tree/DSU Tree Tutorial - Codeforces [138]
- Codeforces [139]
- Mzhang2021's Blog [140]

[129] IOI: Kỳ thi Olympic Tin học Quốc tế.
[130] https://ioi2018.jp/wp-content/tasks/contest1/werewolf.pdf
[131] http://www.usaco.org/index.php?page=viewproblem2&cpid=384
[132] https://www.codechef.com/problems/TULIPS
[133] https://www.codechef.com/problems/CHEFCOMP
[134] APIO: Kỳ thi Olympic Tin học Châu Á -– Thái Bình Dương.
[135] https://tlx.toki.id/problems/apio-2020/B
[136] https://codeforces.com/contest/1706/problem/E
[137] https://codeforces.com/problemset/problem/1628/E
[138] https://codeforces.com/blog/entry/85714
[139] https://codeforces.com/blog/entry/71568
[140] https://mzhang2021.github.io/cp-blog/kruskal/

“Nhảy nhị phân” với bộ nhớ $\mathcal{O}(n)$

Ngô Nhật Quang
12A1 Tin, Trường THPT Chuyên Khoa học Tự nhiên, Đại học Quốc gia Hà Nội



Giới thiệu

Trước khi đọc bài viết bài, người viết giả dụ bạn đọc đã đọc và hiểu về binary lifting [141] (Nâng nhị phân).

Bài viết này sẽ giới thiệu một thuật toán được công bố năm 1983 bởi tiến sĩ Eugene W. Myers. Như thuật toán binary lifting đã được giới thiệu trước trên VNOI Wiki, thuật toán giới thiệu trong bài này cũng có độ phức tạp truy vấn là $\mathcal{O}(\log n)$. Tuy nhiên, với thuật toán này, ta có thể thêm hoặc bớt một lá trong cây trong $\mathcal{O}(1)$, tức tổng bộ nhớ cần thiết chỉ là $\mathcal{O}(n)$ thay vì $\mathcal{O}(n \log n)$.

Lưu ý: Các đoạn code mẫu chỉ dùng để tham khảo chứ không thể biên dịch, ban đọc có thể tham khảo code trong phần Thử nghiệm thời gian chạy.

Cấu trúc

Cơ bản nhất, cấu trúc cây được cài đặt như sau với ba biến cho mỗi đỉnh trên cây:

```
class Node {
  int depth;
  Node parent;
  Node jump;
}
```

Chức năng của `id`, `depth` và `parent` ta có thể đoán được qua tên. Biến `id` lưu trữ số thứ tự của đỉnh này, `depth` sẽ lưu trữ độ sâu của đỉnh này, còn biến `parent` sẽ là con trỏ trỏ đến cha của đỉnh này. Biến `jump` lúc này ta sẽ "tạm" định nghĩa nó là một con trỏ trỏ đến một đỉnh tổ tiên bất kì. Ta sẽ đi sâu hơn việc cài đặt biến `jump` ở phần sau. Biến `root` sẽ là gốc của cây.

Với đỉnh 0 là đỉnh gốc của cây, ta định nghĩa

```
root.parent = null;
root.depth = 0;
root.jump = root;
```

Binary Lifting

Với định nghĩa trên, ta đã có thể dựng ra một đoạn code để tìm tổ tiên có độ sâu là d của của đỉnh u.

```
Node find(Node u, int d) {
  // Lặp đến khi độ sâu đỉnh hiện tại là d
  while (u.depth > d) {
    // Nếu đỉnh jump có độ sâu
    // bé hơn d thì ta chỉ nhảy lên cha
    if (u.jump.depth < d) {
      u = u.parent;
    } else {
      // Còn không ta sẽ nhảy lên qua con trỏ jump
      u = u.jump;
    }
  }
  return u;
}
```

Đoạn code khá đơn giản vì ta cũng khó làm được gì nhiều với hai con trỏ lên trên. Ta sẽ đi lên trên bằng cách "nhảy" liên tục lên đỉnh `jump` mà kiểm tra rằng ta không nhảy quá đỉnh mà ta đang cần tìm.

Định nghĩa con trỏ `jump`

Rõ ràng, với đoạn code như trên, ta đến được đỉnh cần tìm hoàn toàn phụ thuộc vào việc các biến `jump` đưa ta đến đỉnh cần tìm nhanh như nào. Hàm thêm một lá vào cây trông như sau:

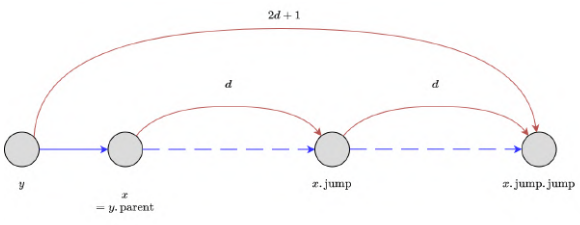
```
Node makeLeaf(Node p) {
  Node leaf = new Node();
  leaf.parent = p;
  leaf.depth = p.depth + 1;

  if (p.depth - p.jump.depth ==
      p.jump.depth - p.jump.jump.depth) {
    leaf.jump = p.jump.jump;
  } else {
    leaf.jump = p;
  }

  return leaf;
}
```

Phần gán biến `id`, `parent` và `depth` khá dễ hiểu, nhưng còn đoạn này thì nghĩa là gì?

```
p.depth - p.jump.depth ==
  p.jump.depth - p.jump.jump.depth
```

Chứng minh tính đúng đắn của nó khá dài, bạn đọc muốn tìm hiểu có thể đọc bài nghiên cứu của tiến sĩ Myers có tên là "An applicative random-access stack [142] ".

Dưới đây sẽ là một số hình ảnh để bạn đọc có thể có một số ý tưởng tại sao nó lại đúng.

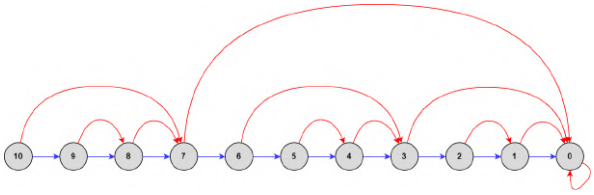


[141] https://vnoi.info/wiki/algo/data-structures/lca-binlift.md
[142] http://myerslab.mpi-cbg.de/wp-content/uploads/2014/06/applicative.stack_.pdf

Ở đây, đỉnh y là lá ta vừa thêm vào cây. Các mũi tên màu đỏ thể hiện các cú nhảy qua con trỏ `jump`, còn mũi tên màu xanh là đi lên cha nó qua con trỏ `parent`. Dễ thấy khi xem lại hàm `find` của ta, nó sẽ thử nhảy bước $2d + 1$, nếu không được thì sẽ nhảy lên đỉnh cha. Bước nhảy ở đỉnh cha sẽ luôn luôn ngắn hơn bước nhảy của đỉnh con, ngắn hơn ít nhất 2 lần.

Nhớ lại, đây không phải giống hệt binary lifting mà ta biết hay sao? Chỉ là thay vì nhồi $\log$ bước nhảy vào trong một đỉnh, ta dàn đều nó ra khắp các đỉnh tổ tiên của nó.

Điều này giúp ta giải thích việc ta luôn tìm được đỉnh có bước nhảy bé hơn nhanh. Thế nếu ta ở một đỉnh rất sâu, mà nó lại không có bước nhảy đủ to thì sao? Vậy thì việc có bước nhảy bé hơn thì có tác dụng gì? Ta nhìn vào bức ảnh lớn hơn:



Bức ảnh không lớn lắm, nhưng mong bạn đọc có thể nhận ra bức tranh tổng thể mà thuật toán muốn thực hiện. Nhận thấy rằng sau tối đa hai bước nhảy qua con trỏ `jump`, ta sẽ đến được một đỉnh có bước nhảy lớn hơn ít nhất gấp đôi. Tức là, nếu ta cứ nhảy qua con trỏ `jump` liên tục, độ dài của bước nhảy sẽ tăng theo cấp số nhân.

Qua hai bức ảnh này, ta có thể phần nào đó nhận ra cách mà hàm `find` hoạt động:

- Tăng tốc (tìm bước nhảy lớn hơn) bằng cách nhảy liên tục
- Giảm tốc (tìm bước nhảy ngắn hơn) bằng cách sử dụng bước nhảy của cha nó.

Rõ ràng, lúc này ta có thể thấy ta không thể đảm bảo sử dụng tối đa $\log$ bước nhảy để đến đích. Bài nghiên cứu của tiến sĩ Myers cho ta cận trên $3\lfloor \log_2 (d+1) \rfloor - 2$ với d là độ sâu của đỉnh khởi đầu.

Thử nghiệm thời gian chạy

Như phần trước, ta có được cận trên của thuật toán là sử dụng $3\lfloor \log (d+1) \rfloor - 2$. Vậy có nghĩa là thực tế code ta sẽ chạy lâu gấp 3 lần code binary lifting bình thường đúng không?

~~Nếu đúng thì đã không có phần này.~~ Ta xem các submission sau đây:

Link bài	Bộ nhớ $\mathcal{O}(n)$	Bộ nhớ $\mathcal{O}(n \log n)$
VNOJ - Sloth Naptime[143]	441ms[144]	1265ms[145]
Codeforces - Minimum spanning tree for each edge[146]	265 ms[147]	358 ms[148]
Library Checker - Lowest Common Ancestor[149]	436 ms[150]	664 ms[151]

Điều này cho ta thấy, ít nhất là trên các bộ test ngẫu nhiên, thì binary lifting bộ nhớ $\mathcal{O}(n)$ chạy nhanh hơn, thậm chí là nhanh gấp đôi, gấp ba lần. Theo suy đoán của tác giả, điều này liên quan đến cách sử dụng bộ nhớ đệm của mỗi thuật toán. Bạn đọc muốn tìm hiểu thêm có ở đọc tại đây [152] . Đây cũng là lí do mà tại sao quy hoạch động cuốn chiếu không chỉ dùng ít bộ nhớ hơn mà còn thường chạy nhanh hơn quy hoạch động thông thường.

Tìm tổ tiên chung sâu nhất của hai đỉnh

Như cách ta định nghĩa con trỏ `jump` như trên, ta có thể thấy rằng với hai đỉnh bất kì với cùng độ sâu, thì hai đỉnh `jump` của chúng cũng sẽ có độ sâu giống nhau. Do vậy, giả dụ hai đỉnh được cho là u và v và đỉnh sâu hơn là u, thì ta tìm đỉnh s là tổ tiên của đỉnh u có độ sâu bằng đỉnh v. Sau đó, ta nhảy lên dần dần ở cả hai đỉnh cho đến khi chúng giống nhau.

[143] https://oj.vnoi.info/problem/secondthread_tree_sloth
[144] https://oj.vnoi.info/submission/2147925
[145] https://oj.vnoi.info/submission/336556
[146] https://codeforces.com/contest/609/problem/E
[147] https://codeforces.com/contest/609/submission/189027148
[148] https://codeforces.com/contest/609/submission/136238260
[149] https://judge.yosupo.jp/problem/lca
[150] https://judge.yosupo.jp/submission/120907
[151] https://judge.yosupo.jp/submission/120909
[152] https://stackoverflow.com/questions/16699247/what-is-a-cache-friendly-code

```
Node lca(Node u, Node v) {
  if (u.depth < v.depth) swap(u, v);
  u = find(u, v.depth);

  while (u != v) {
    if (u.jump == v.jump) {
      u = u.parent;
      v = v.parent;
    } else {
      u = u.jump;
      v = v.jump;
    }
  }

  return u;
}
```

Tham khảo

- "Binary Lifting, No Memory Wasted [153] " - Urbanowicz

[153] https://codeforces.com/blog/entry/74847

Ban biên tập

Nguyễn Trung Quân (tổng biên tập)

Võ Nguyễn Phương Quỳnh

Phạm Thị Diễm Quỳnh

Mai Vinh Hiển

Thiết kế

Nguyễn Trúc Như Bình

Trần Quang Lộc

Hình ảnh sử dụng

Các hình ảnh được sử dụng để thiết kế nền các trang được lấy từ trang web Flaticon

- Children toys icons created by Freepik
- Origami icons created by photo3idea_studio
- Cactus icons created by smalllikeart
- Book icons created by itim2101
- Cd icons created by geotatah

From VNOI with ♥