

THE ICPC 2018 VIETNAM CENTRAL PROVINCIAL CONTEST

DANANG UNIVERSITY OF SCIENCE AND TECHNOLOGY
OCTOBER 14, 2018

Editorial











Tổng quan

Bài	First AC	#AC	Editorialist
A	4 HCMUS-Intimidate & HCMIU	172	Nguyễn Khánh
B	10 hashset	95	Phạm Tuấn Nghĩa
C	206 bitset	3	Nguyễn Thành Vinh
D	36 THREE	86	Lê Minh Quang
E	22 HCMUS-Intimidate & CHY.KB	126	Hoàng Xuân Nhật
F	64 FPTU Team1	43	Nguyễn Diệp Xuân Quang
G	45 THREE	25	Vương Hoàng Long
H	92 vector	3	Lăng Trung Hiếu
I	199 HCMUS-Ascension	2	Nguyễn E Rô
J	12 map	180	Nguyễn Diệp Xuân Quang
K		0	Lăng Trung Hiếu & Nguyễn Ngọc Trung
L	137 THREE	5	Lê Anh Đức

Bảng xếp hạng chung cuộc

The ICPC 2018 - Vietnam Central Provincial Contest

Final standings

Standings																
#	Who		=	Penalty	A	B	C	D	E	F	G	H	I	J	K	L
1		THREE VNU University of Science	9	813	1 9	2 70	0	1 36	1 34	1 152	2 45	0	1 209	1 81	0	1 137
2		bitset University of Engineering and Technolo	9	925	1 21	1 25	1 206	1 67	1 44	2 86	1 131	1 1	1 1	2 37	1 1	2 248
3		amazingbamboo_with_cocco Hanoi University of Science and Techno	9	965	2 32	1 26	0	1 70	1 56	1 77	2 102	2 256	0	1 38	0	1 248
4		map University of Engineering and Technolo	9	1172	1 51	1 29	4 295	1 65	1 46	2 99	2 199	1 1	0	1 12	0	1 276
5		vector University of Engineering and Technolo	8	758	2 28	2 41	0	1 76	2 49	3 159	1 157	1 92	1 1	1 56	0	0
6		HCMUS-Ascension University of Science, VNU-HCM	8	860	1 27	1 14	0	1 65	3 68	5 111	1 221	0	1 199	1 35	1 1	0
7		HCMUS-Intimidate University of Science, VNU-HCM	7	459	1 4	1 13	0	1 50	1 22	1 85	5 178	2 1	1 1	1 27	0	0
8		Ams.Fusion Hanoi University of Science and Techno	7	526	1 7	1 38	0	1 45	1 32	1 129	5 143	0	0	2 32	1 1	0
9		multimap University of Engineering and Technolo	7	582	2 21	2 49	2 1	2 90	2 81	1 130	2 71	8 1	0	1 40	0	0
10		ONE VNU University of Science	7	615	1 14	2 31	0	1 53	1 80	8 1	0	2 104	0	2 62	0	1 211

A

Lời giải: Nguyễn Khánh (map)

Ta sẽ sử dụng một đẳng thức quen thuộc để giải quyết bài này, đó là:

$$|a| + |b| = \max(\pm a \pm b)$$

Do đó:

$$|x_i - x_j| + |i - j| = \max(\pm(x_i - x_j) \pm (i - j))$$

Hay:

$$\begin{aligned} |x_i - x_j| + |i - j| = \max\{ & (x_i + i) - (x_j + j), \\ & (x_i - i) - (x_j - j), \\ & -(x_i + i) + (x_j + j), \\ & -(x_i - i) + (x_j - j) \} \end{aligned}$$

Từ đó ta có thể thấy rằng kết quả bài toán sẽ là max của **2** trường hợp đầu với mọi i, j .

Với mỗi trường hợp, bài toán trở thành: cho một mảng, tính hiệu của phần tử lớn nhất và phần tử bé nhất. Ta dễ dàng giải quyết bài toán bằng cách duyệt qua cả mảng, tìm hai phần tử lớn nhất và nhỏ nhất xong lấy hiệu.

ĐPT: $O(n)$

B

Lời giải: Phạm Tuấn Nghĩa (bitset)

- Thuật toán $O(N^3)$: Ta nhận thấy chỉ cần duyệt vị trí bắt đầu của các substring và quét từ trái qua phải, duy trì “richness” của các substring bằng cách đánh dấu các ký tự đã xuất hiện. Qua đó, ta có thể so sánh các substring với kết quả đã tìm được và chọn xâu tối ưu hơn. Thuật toán trên yêu cầu chúng ta phải so sánh xâu kết quả với tất cả substring nên sẽ không đáp ứng time limit.
- Thuật toán $O(N^2)$: Từ ý tưởng trên, thay vì mỗi lần so sánh 1 substring với kết quả thì ta có thể so sánh các substring với nhau trước rồi mới cập nhật kết quả. Dễ dàng nhận thấy, với các substring bắt đầu ở một vị trí P, ta chỉ cần chọn substring ngắn nhất có “richness” lớn nhất do prefix của một xâu luôn có thứ tự từ điển nhỏ hơn xâu đó.

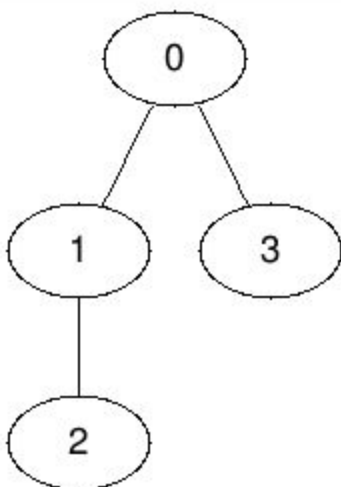
C

Lời giải: Nguyễn Thành Vinh (bitset)

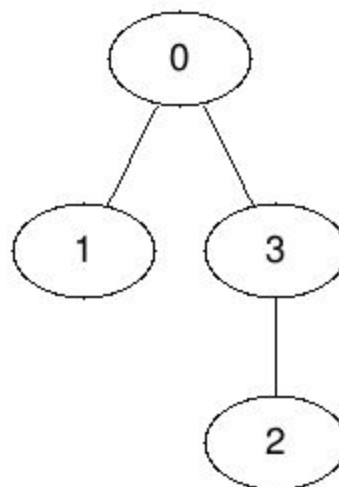
Hiển nhiên trong bài toán này đáp án nhỏ nhất có thể là 0 (khi tập đỉnh ta chọn là tập rỗng).

Giả sử kết quả không phải là 0 (phải chọn ít nhất một đỉnh), ta sẽ chọn một đỉnh thuộc tập sẽ chọn (gọi là *root*). Như vậy bài toán đưa về chọn một subgraph có chứa đỉnh *root* ở cả hai cây sao cho tổng trọng số $score[i]$ là lớn nhất.

Sau đó DFS lại hai cây sao cho gốc của cả hai cây là *root*. Xét test đề, nếu gốc là 0, ta có hai cây sau:



Cây 1



Cây 2

Theo giả thuyết của chúng ta, 0 đã được chọn, bây giờ nếu muốn chọn thêm nút 2, thì:

- Trong cây 1, ta phải chọn thêm nút 1, vì nếu không chọn nút 1 thì sẽ mất liên thông của nút gốc và nút 2.
- Trong cây 2, ta phải chọn thêm nút 3, vì nếu không chọn nút 3 thì sẽ mất liên thông của nút gốc và nút 3,

Như vậy, với hai cây đã cho và đỉnh *root*, ta sẽ có một số ràng buộc: **"nếu chọn *u* thì phải chọn *v*", cụ thể là nếu chọn một đỉnh khác gốc thì ta cũng phải chọn cả cha của nó**, và yêu cầu đề bài sẽ là chọn tập đỉnh có trọng số $score[i]$ là lớn nhất.

Đây chính là bài luồng tập đóng, có xuất hiện trong VOI 2014 (<http://vnoi.info/problems/JOBSET/>).

Lời giải cụ thể của bài JOBSET các bạn có thể đọc ở:

http://vnoi.info/problems/list_solutions/JOBSET/

hoặc có thể tham khảo ở link Tiếng Anh (về Project Selection Problem):

<http://www14.in.tum.de/lehre/2015WS/ea/split/sub-Project-Selection.pdf>

D

Lời giải: Lê Minh Quang (amazingbamboo_with_coccoc)

Gọi s_i là [tổng cộng dồn](#) của các giá trị a_j với $j \leq i$.

Gọi f_i là số cách để xếp s_i quả bóng với i màu đầu tiên thỏa mãn điều kiện: quả cuối của màu j đứng trước quả cuối của màu $j + 1$ với $1 \leq j < i$.

Ta có:

$$f_i = g_i * f_{i-1}$$

Với g_i là số cách để xếp a_i quả bóng màu i trong tổng số s_i vị trí. Để điều kiện thỏa mãn, quả cuối cùng trong số s_i quả phải là màu i . Còn lại $a_i - 1$ quả, ta có thể tự do xếp chúng vào

$s_i - 1$ vị trí trống với số cách xếp là $\binom{s_i - 1}{a_i - 1}$. Như vậy:

$$g_i = \binom{s_i - 1}{a_i - 1}$$

$$f_i = g_i * f_{i-1} = \binom{s_i - 1}{a_i - 1} * f_{i-1}$$

Để ý rằng $s_i \leq 1000$, vậy nên ta hoàn toàn có thể chuẩn bị giá trị của các tổ hợp theo công thức:

$$\binom{n}{k} = \binom{n - 1}{k - 1} + \binom{n - 1}{k}$$

Kết quả bài toán là f_k .

Độ phức tạp thuật toán: $O(k)$.

E

Lời giải: Hoàng Xuân Nhật (HCMUS-Ascension)

Chúng ta cần tìm một tập lớn nhất các hình tròn sao cho khoảng cách giữa hai hình tròn bất kì lớn hơn tổng bán kính của chúng. Tuy nhiên, trong bài này, các hình tròn nằm trên một đường thẳng nên ta có thể coi mỗi hình tròn như một đoạn thẳng từ $[x_i - w_i, x_i + w_i]$. Khi đó bài toán chuyển thành tìm một tập các đoạn thẳng sao cho hai đoạn bất kì thì không giao nhau (có thể chạm nhau tại hai đầu mút). Bài toán này có thể giải quyết dễ dàng bằng cách [rời rạc hóa](#) các điểm đầu mút lại sau đó quy hoạch động với dp_i là kết quả của bài toán cho i điểm đầu tiên. Độ phức tạp là $O(n \log n)$.

F

Lời giải: Nguyễn Diệp Xuân Quang (HCMUS-Intimidate)

Nhận xét rằng, khi hai con kiến gặp nhau, thay vì cho chúng quay đầu lại, ta có thể xem như chúng nhảy qua nhau và trao đổi số thứ tự cho nhau. Nói cách khác, gọi:

- x_i là thời điểm mà con kiến thứ i rời khỏi đoạn dây nếu ta xét sự tương tác giữa các con kiến
- y_i là thời điểm mà con kiến thứ i rời khỏi đoạn dây nếu ta không xét sự tương tác giữa các con kiến

Thì sẽ tồn tại một hoán vị p sao cho $x_{p_i} = y_i$. Do đó, $\max(x_1, x_2, \dots, x_N) = \max(y_1, y_2, \dots, y_N)$. Ta dễ dàng tính được giá trị y_i như sau:

- Nếu con kiến thứ i bò về hướng bên trái thì $y_i = a_i$
- Nếu con kiến thứ i bò về hướng bên phải thì $y_i = L - a_i$

Bài toán ban đầu trở thành:

Cho một dãy số a_i gồm N phần tử và số L . Ta xây dựng dãy số b bằng cách, với mỗi phần tử b_i , ta sẽ gán ngẫu nhiên giá trị a_i hoặc $L - a_i$ với xác suất như nhau. Tính giá trị kì vọng của phần tử lớn nhất trong dãy b .

Để tính giá trị kì vọng trên, với mỗi giá trị x , ta tính prob_x là xác suất để x trở thành phần tử lớn nhất trong dãy b . Khi đó, đáp số là tổng của các tích $x * \text{prob}_x$.

Trước hết, ta sẽ rút gọn dãy a bằng cách xét phần tử a_i có giá trị lớn hơn $L/2$:

- Nếu tồn tại p sao cho $a_p = L - a_i$ thì ta xem như b_i được gán giá trị a_i với xác suất 25% và $L - a_i$ với xác suất 75%. (và bỏ qua phần tử b_p khi tính giá trị lớn nhất).
- Ngược lại, ta gán $a_i = L - a_i$, và điều này sẽ không làm thay đổi đáp án.

Sau đó, ta sắp xếp dãy a theo thứ tự tăng dần. Gọi p_i là xác suất để b_i được gán giá trị a_i . Ta nhận xét, phần tử lớn nhất trong dãy b chỉ có thể là a_N hoặc $L - a_i$ với $1 \leq i \leq N$.

Để $L - a_i$ trở thành phần tử lớn nhất dãy b thì:

- Với $1 \leq j < i$ thì b_j phải được gán giá trị a_j
- Bản thân b_i phải được gán giá trị $L - a_i$

Do đó, $\text{prob}_{L-a_i} = p_1 p_2 \dots p_{i-1} * (1 - p_i)$.

Để a_N trở thành phần tử lớn nhất dãy b thì tất cả các phần tử b_i đều phải được gán giá trị a_i . Do đó, $\text{prob}_{a_N} = p_1 p_2 \dots p_N$.

Ta có thể dễ dàng tính các giá trị trên bằng một vòng lặp for trong $O(N)$. Do chi phí sắp xếp nên độ phức tạp là $O(N \log A_i)$.

G

Lời giải: Vương Hoàng Long (map)

Bài này thực ra là một bài khá đơn giản, nhưng lại có thể làm bối rối các team không biết chia trong base khác 10.

Trước tiên, ta sẽ viết lại cách biểu diễn 1 số ở base khác dưới dạng như sau:

$$X = x_1 * base^n + x_2 * base^{n-1} + \dots + x_n * base^0$$

$$Y = y_1 * base^n + y_2 * base^{n-1} + \dots + y_n * base^0$$

Vậy nếu muốn chia X cho Y trong base khác 10, ta chỉ cần chia 2 số trong base 10 như bình thường, và sau đó chuyển kết quả sang base tương ứng. (Giống hệt như chia đa thức). Nhân 2 số trong base khác 10 chúng ta cũng làm như vậy.

Trước tiên, ta sẽ giải bài toán ở base 10:

Ở đây mình sẽ ví dụ việc tìm chữ số thứ 5 sau dấu “,” của phép chia 5/19 trong base 10

$$\text{Ta có } 5/19 = 0, 5*10/19 = 2 \Rightarrow 12*10/19 = 6 \Rightarrow 6*10/19 = 3 \Rightarrow 3*10/19 = 1 \Rightarrow 11*10/19 = 5$$

Vậy 5/19 = 0,26315 => chữ số thứ 5 là 5

Ta nhận xét thấy để tìm chữ số thứ k sau dấu “,” , ta có thể nhân $10^{(k-1)}$ số bị chia lên và chuyển bài toán thành tìm chữ số thứ 1 sau dấu “,”.

Gọi số bị chia là A, số chia là B.

Để giải được bài toán tìm chữ số thứ 1 sau dấu “,” , ta sẽ tìm dư của $A*10^{(k-1)} \% B$, sau đó kết quả của bài toán sẽ là thương của $(\text{số dư} * 10 / B)$. Để tìm được số dư của phép $A*10^{(k-1)} \% B$, ta có thể sử dụng kĩ thuật nhân ẩn độ do A và B đều là 10^{18} . Tham khảo:

<https://cowboycoder.tech/article/phep-nhan-an-do-va-phep-tinh-luy-thua>

Để tìm được thương của $(\text{số dư} * 10 / B)$, ta sẽ phải sử dụng Bignum chia.

Vậy nếu muốn giải bài toán ở base khác thì sao? Rất đơn giản, các bạn có thể nhìn vào cách đặt phép chia 5/19 ở trên và nhận ra thay vì nhân số dư với 10, ta nhân số dư với BASE.

Vậy solution của bài toán chuyển thành: nhân $BASE^{(k-1)}$ số A lên, tìm số dư của phép chia $(BASE^{(k-1)})$ cho B và kết quả bài toán sẽ là thương của $(\text{số dư} * \text{base})/B$. Phương pháp giải vẫn sử dụng nhân Ẩn Độ kết hợp Bignum chia. Chú ý, vì mọi tính toán của mình đều được thực hiện ở base 10 nên kết quả cũng đã là base 10, không cần chuyển sang base 10 khi in ra nữa.

Có rất nhiều đội sử dụng Python để giải bài toán này, đó là một cách rất hiệu quả để tránh phải cài đặt nhân Ẩn Độ và Bignum chia trong C++, hơn nữa code cũng ngắn hơn rất nhiều

H

Lời giải: Lăng Trung Hiếu

Với $n = 2$ chính là bài game này [Wythoff's game](#). Có thể dễ dàng tìm được các cặp người 2 thắng (1,2) (3,5) (4,7) (6,10) bằng cách tìm số nguyên dương đầu tiên chưa xuất hiện X và cặp ở vị trí thứ i sẽ là $(X, X + i)$.

Với $n > 2$, có một đoạn khá nhỏ trong input "prime" tức n là số nguyên tố hay n lẻ. Có thể dùng dp trạng thái để tìm ra quy luật: đặt $S = \text{xor}(a[i])$ ($i = 1, 2, \dots, n$) thì nếu S khác 0 người 1 thắng, $S = 0$ người 2 thắng.

Chứng minh: Theo như bài toán bốc sỏi thông thường thì người có S khác 0 luôn có cách để bốc về $S = 0$ và người có $S = 0$ không thể bốc từ 1 đồng để về $S = 0$ (mà trạng thái cuối cùng của trò chơi là $S = 0$). Vậy để người có $S = 0$ bốc về một trạng thái $S' = 0$ thì phải bốc tất cả các đồng sỏi một lượng là x viên. Giả sử bit = 1 nhỏ nhất của x là u thì khi đó ở vị trí bit u của $a[i]$ và $a[i] - x$ ($i = 1, 2, \dots, n$) sẽ khác nhau vì vậy bit u của $S' = \text{xor}(a[i] - x)$ ($i = 1, 2, \dots, n$) sẽ = 1 do n lẻ. Vậy tức là người có trạng thái $S = 0$ không thể bốc về một trạng thái $S' = 0$ hay với n lẻ chính là bài toán bốc sỏi thông thường.

Lời giải: Nguyễn E Rô (HCMUS-Ascension)

Chúng ta có 1 bảng $3^t \times 3^t$

Nhận xét: Các dòng có chứa ít nhất 1 ô màu đen thì giống nhau và các cột cũng vậy.

Với hình chữ nhật cần kiểm tra, đầu tiên ta kiểm tra điều kiện trên (các hàng có ít nhất 1 ô màu đen thì giống nhau, các cột có ít nhất 1 ô màu đen cũng giống nhau). Nếu hình chữ nhật kiểm tra không thoả mãn điều kiện này thì kết quả = 0.

Coi mỗi cột có ít nhất một ô màu đen là số 0 và các cột chỉ toàn màu trắng là số 1, chúng ta có một dãy số nhị phân và gọi đây là mảng Column. Tương tự ta cũng có một mảng tương tự là mảng Row (nhưng thực chất 2 mảng này hoàn toàn giống nhau, chỉ gọi 2 cái tên khác nhau cho dễ phân biệt).

Với hình chữ nhật cần kiểm tra, chúng ta cũng tạo ra 2 dãy Col2 và Row2 tương tự như trên. Tới đây chúng ta có thể chứng minh được rằng hình chữ nhật đó nằm trong hình vuông đề bài cho khi và chỉ khi dãy Col2 kia là 1 dãy con gồm các phần tử liên tiếp nhau của dãy Column và dãy Row2 cũng phải là dãy con gồm các phần tử liên tiếp nhau của dãy Row.

Đến đây thì công việc rất đơn giản, đếm xem dãy Col2 xuất hiện bao nhiêu lần trong dãy Column và dãy Row2 xuất hiện bao nhiêu lần trong dãy Row và gọi chúng lần lượt là cntCol và cntRow.

Đáp số của chúng ta cần tìm chính là cntCol * cntRow.

Việc đếm có thể sử dụng KMP, Z Algorithm, Hash, ...

ĐPT $O(3^t)$

Có thể tối ưu bài toán là co các đoạn có số 0 liên tiếp lại với nhau, khi đó số phần tử trong mảng ban đầu tối đa là $2^{t+1} - 1$ phần tử (2^t số 0 đại diện cho các ô có màu đen và $2^t - 1$ ô có giá trị là số lượng ô màu trắng liên tiếp đặt giữa 2 ô màu đen liên tiếp) thì ĐPT bài toán là $O(2^{t+1})$ nhưng cài đặt phức tạp hơn một chút.

J

Lời giải: Nguyễn Diệp Xuân Quang (HCMUS-Intimidate)

Trước hết, ta sắp xếp tọa độ của các bụi cỏ tăng dần.

Nếu gọi x_{\min} và x_{\max} lần lượt là tọa độ nhỏ nhất và lớn nhất trong các bụi cỏ mà ta cắt, thì để đi đến x_{\min} và x_{\max} từ tọa độ 0, ta cần phải đi qua tất cả các bụi cỏ có tọa độ nằm giữa x_{\min} và x_{\max} . Do đó, ta hoàn toàn có thể cắt đi tất cả các bụi cỏ nằm giữa x_{\min} và x_{\max} . Nói cách khác, K bụi cỏ được cắt sẽ nằm liên tiếp nhau.

Ta duyệt i từ 1 đến $N - K + 1$, tìm quãng đường nhỏ nhất để cắt các bụi cỏ từ i đến $i+K-1$ và đáp án sẽ là min của các quãng đường tìm được. Để cắt các bụi cỏ từ i đến $i+K-1$, có hai cách di chuyển sau:

- $0 \rightarrow x_i \rightarrow x_{i+K-1}$: Quãng đường là $|x_i| + |x_{i+K-1} - x_i|$.
- $0 \rightarrow x_{i+K-1} \rightarrow x_i$: Quãng đường là $|x_{i+K-1}| + |x_{i+K-1} - x_i|$.

Do đó, quãng đường nhỏ nhất là $\min(|x_i|, |x_{i+K-1}|) + |x_{i+K-1} - x_i|$.

Độ phức tạp: $O(N \log N)$ do chi phí sắp xếp.

K

Lời giải: Lăng Trung Hiếu

Để làm được bài này cần có những kiến thức cơ bản và nâng cao của hàm sinh, số euler, số phức, tổ hợp, ánh xạ.

Để thấy với $k = 1$ ta có công thức m^n , ta có thể dùng hàm sinh như sau:

- Với một số lượng cố định a_1 số 1, a_2 số 2, a_m số m thì sẽ có $n! / ((a_1)! * (a_2)! * \dots * (a_m)!)$ cách chọn dãy \Rightarrow số cách chọn tổng quát sẽ là hệ số của x^n trong khai triển $F = (1 + x/1! + x^2/2! + x^3/3! + \dots)^m * (n!)$ (giả sử với $m = 3, n = 11$ ai lần lượt là 2,4,5 thì ta có hệ số của x^{11} có chứa $x^2/2! * x^4/4! * x^5/5! * 11!$). Dùng [Euler Formula](#) thì sẽ có $e^x = (1 + x/1! + x^2/2! + \dots)$

$\Rightarrow F = (e^x)^m * n!$ hay $= n! * e^{(x*m)}$ mà dùng taylor lần nữa có $e^{(x*m)} = 1 + (xm)/1! + (xm)^2/2! + \dots + (xm)^k/k! + \dots \Rightarrow$ hệ số của x^n trong F là $m^n / n! * n! = m^n$

- Với $k = 2$ thì ta thấy số lượng các số là chia hết cho 2 nên kết quả chính là hệ số của n trong khai triển $F = n! * (1 + x^2/2! + x^4/4! + \dots + x^{(2k)}/(2k)!)^m$. theo biến đổi dãy taylor thì $e^{(-x)} = (1 - x/1! + x^2/2! - x^3/3! + \dots + (-1)^k * x^k/k! + \dots) \Rightarrow F = n! * ((e^x + e^{-x})/2)^m = n! * (e^x + e^{-x})^m / 2^m$. Để tính $(e^x + e^{-x})^m$ thì có thể for số lượng e^x là a , số lượng e^{-x} là $m - a \Rightarrow$ số mũ là $x^a + -x(m - a) = x(2a - m)$ và hệ số là tổ hợp chập a của m phần tử $= C(m, a)$. Tiếp tục tính hệ số của n trong $e^{(kx)} = k^n/n!$ ($n!$ có thể triệt tiêu với $n!$ đầu tiên trong F nên khá dễ dàng để tính)

- Khoan nếu vậy để tính $k = 3$ thì rất khó nên ta đi tìm đặc điểm tại sao $k = 2$ thì ta lại có $(e^x + e^{-x})/2$. Từ khai triển taylor $e^{(kx)} = 1 + (kx)/1! + (kx)^2/2! + \dots + (kx)^t/t! + \dots$ từ đây ta thấy ta có thể giải một phương trình với k_1, k_2 sao cho $(k_1 + k_2) = 0$ và $(k_1^2 + k_2^2) = 2$ khi đó sẽ có $e^{(k_1x)} + e^{(k_2x)} = 2 + 0 + 2*x^2/2! + 0 + 2*x^4/4! + \dots + 2*x^{(2k)}/(2k)! + \dots = 2 * (1 + x^2/2! + x^4/4! + \dots + x^{(2k)}/(2k)! + \dots)$

Từ hệ trên ta thấy k_1, k_2 là nghiệm của phương trình $(x^2 - 1)$ (bạn đọc tự chứng minh) $\Rightarrow k_1, k_2$ là 1 và -1. Tương tự như vậy cho $k = 3$. Nếu ta đặt $(k_1 + k_2 + k_3) = 0, (k_1^2 + k_2^2 + k_3^2) = 0, (k_1^3 + k_2^3 + k_3^3) = 3$ thì ta sẽ có k_1, k_2, k_3 là nghiệm của phương trình $(x^3 - 1) = 0$ (cái này phải khai triển ra sẽ thấy) hay $k_1 = 1, k_2 = (\sqrt{-3} - 1)/2, k_3 = (-\sqrt{-3} - 1)/2$. Với $k = 4$, ta đi giải nghiệm của phương trình $x^4 - 1 = 0$ thì sẽ được 4 nghiệm 1, -1, $i, -i$

- Với $k = 3$ bài toán đặt ra là đi tìm hệ số của x^n trong khai triển của $(e^{(k_1x)} + e^{(k_2x)} + e^{(k_3x)})^m$ với $k_1 = 1, k_2 = (\sqrt{-3} - 1)/2, k_3 = (-\sqrt{-3} - 1)/2$. Ta thấy nếu đặt số lượng bằng $e^{(k_1x)} = a, e^{(k_2x)} = b$ thì số lượng bằng $e^{(k_3x)} = m - a - b$ và hệ số là $m! / (a! * b! * (m - a - b)!)$ và số mũ là $e^{(k_1*a + k_2*b + k_3 * (m - a - b))}$. Theo khai triển taylor số mũ của x^n trong $(e^{(dx)})$ là $d^n / n!$ nên ta cần tính $(k_1*a + k_2*b + k_3 * (m - a - b))^n$. Có thể đặt $(k_1*a + k_2*b + k_3 * (m - a - b))$ dưới dạng $u + v*\sqrt{3}*i$ ta cần tìm $(u + v*\sqrt{3}*i)^n$. Ta có $(a +$

$b\sqrt{3}^i * (c + d\sqrt{3}^i) = (ac + 3bd + (ad + bc)\sqrt{3}^i) = (A + B\sqrt{3}^i)$ từ đây ta thấy có thể dễ dàng tính $(u + v\sqrt{3}^i)^n$ bằng cách mũ hoá thông thường mỗi lần lưu lại hai giá trị u, v trong biểu thức (u là hệ số thực còn $v * \sqrt{3}$ là hệ số phức)

- Tương tự với $k = 4$ ta có khai triển $(e^x + e^{-x} + e^i + e^{-i})^m$, ta có thể lựa chọn a bằng e^x , b bằng e^{-x} , c bằng e^i và d bằng e^{-i} với $a + b + c + d = m$ khi đó hệ số là $m!/(a! * b! * c! * d!)$ và số mũ là $e^{(a - b + (c - d) * i)}$. Nếu làm như vậy thì ta sẽ mất độ phức tạp là khoảng $C(3, m) * \log_2(10^9)$ xấp xỉ 3 tỷ (chọn a, b, c sau đó tính $d = m - a - b - c$ và nhân với cái mũ hoá số phức) sẽ bị TLE

- Vậy phải làm sao với $k = 4$. Ta thấy khi mũ số phức ta chỉ cần quan tâm đến $a - b$ và $c - d$. Nếu ta đặt $a + c = u$ và $a + d = v$, vì $a + b + c + d = m \Rightarrow a - b = (u + v - m)$ và $c - d = u - v$ với $0 \leq u \leq m$ và $0 \leq v \leq m$. Vậy thay vì for 3 vòng a, b, c ta chỉ cần for 2 vòng u, v và hệ số mỗi lần lựa chọn là tổ hợp chập u của m nhân với tổ hợp chập v của m . Để dễ hiểu ta gọi tập A, B, C, D lần lượt là tập gồm a, b, c, d phần tử thuộc m với A hợp B hợp C hợp $D = S$ chứa m phần tử và A, B, C, D đôi một không có phần tử chung. gọi $U = A$ hợp C , $V = A$ hợp D thì với mỗi A, B, C, D sẽ xác định duy nhất một U, V và với mỗi U, V sẽ xác định duy nhất một A, B, C, D ($A = U$ giao V , $B = S - U - V$, $C = U - V$, $D = V - U$). Như vậy có một song ánh từ A, B, C, D sang U, V hay số cách tính số tập A, B, C, D chính là số cách tính số tập U, V .

Lời giải: Nguyễn Ngọc Trung

Bài toán có thể phát biểu lại dưới dạng đơn giản như sau: “Đếm số dãy (có thứ tự) gồm n phần tử mà mỗi phần tử có thể từ 1 đến m và số lượng mỗi số từ 1 đến m xuất hiện trong dãy là một bội của k ”.

Giả sử ta đã xác định được số lượng xuất hiện của i là a_i , thì số dãy sinh ra là $n! / a_1! / a_2! / \dots / a_m!$.

Do đó số lượng dãy cần tìm là hệ số của x^n của hàm sinh:

$$n!(1 + x^k / k! + x^{2k} / (2k)! + x^{3k} / (3k)! + \dots)^m. (*)$$

Chuỗi này rất giống với chuỗi của hàm số $e^x = 1 + x / 1! + x^2 / 2! + x^3 / 3! + \dots$, nhưng bị lược bỏ đi số hạng có số mũ không chia hết cho k . Để giải quyết vấn đề này ta phải sử dụng đến số phức. (Bạn nào học số phức lớp 12 có thể đã gặp bài tính $C(0, n) + C(k, n) + C(2k, n) + \dots$, ý tưởng hoàn toàn tương tự như thế)

Gọi u là k -th root của đơn vị, ta có tính chất sau đây:

- $u^{0i} + u^{1i} + u^{2i} + \dots + u^{(k-1)i} = k$, với $i \% k = 0$.
- $u^{0i} + u^{1i} + u^{2i} + \dots + u^{(k-1)i} = 0$, với $i \% k = 1, 2, \dots, k-1$.

Chú ý thêm là $e^{cx} = 1 + cx / 1! + c^2x^2 / 2! + c^3x^3 / 3! + \dots$, nên có thể dễ dàng thấy được hàm sinh (*) trở thành: $n!(e^x + e^{ux} + e^{u^2x} + \dots + e^{u^{(k-1)}x})^m / k^m$.

Với k cụ thể từ 2 đến 4 ta sẽ cần phải xử lý các hàm sinh sau:

- $k = 2$: $n!(e^x + e^{-x})^m / 2^m$.
- $k = 3$: $n!(e^x + e^{ux} + e^{u^2x})^m / 3^m$, trong đó u là 2th root của đơn vị, $u = (\sqrt{-3} - 1) / 2$.

3. $k = 4: n!(e^x + e^{-x} + e^{ix} + e^{-ix})^m / 4^m.$

Ta sẽ khai triển hàm sinh để lấy tất cả số hạng dạng e^d , sau đó việc còn lại là tính hệ số của x^n của e^d và nó chính là $d^n / n!$. (hiển nhiên $n!$ sẽ bị triệu tiêu nên ta không cần quan tâm)

1. Trường hợp $k = 2$, khá đơn giản, dùng khai triển Newton, ta còn có thể giải cho m lên tới 10^6 .
2. Trường hợp $k = 3$, ta cũng có thể dùng khai triển Newton cho 3 số trong $O(m^2)$, nhưng rắc rối ở đây là dạng của số d trong số hạng e^d tìm được. Nó sẽ có dạng $a + b.\sqrt{-3}$. Để vượt qua, ta có thể định nghĩa thêm một kiểu số $Z[\sqrt{-3}]$ (tập hợp tất cả các số có dạng $a + b.\sqrt{-3}$; a, b là số nguyên) và các phép toán như $+$, $-$, $*$, $/$ có thể thực hiện trên $Z[\sqrt{-3}]$. Khi đó $(a + b.\sqrt{-3})^n$ có thể tính tương tự như tính lũy thừa cho số nguyên và sẽ có dạng $c + d.\sqrt{-3}$.
3. Trường hợp $k = 4$, dùng khai triển Newton cho 4 số sẽ có độ phức tạp là $O(m^3)$. Nhưng d có dạng $a + bi$, với $-m \leq a, b \leq m$ nên sẽ có $O(m^2)$ số d phân biệt. Ta có thể chạy $O(m^3)$ khai triển Newton (với hệ số khá nhỏ) để tìm số lượng bằng $a + bi$. Một cách khác là biến đổi $(e^x + e^{-x} + e^{ix} + e^{-ix})^m = (e^x + e^{ix})^m (1 + e^{-x - ix})^m$. Dùng khai triển Newton cho 2 số cho từng vế, ta sẽ liệt kê được tất cả các số hạng của hàm sinh trong $O(m^2)$. Cũng giống như trường hợp $k = 3$, ta định nghĩa thêm kiểu số $Z[i]$ (tập hợp tất cả các số có dạng $a + bi$; a, b là số nguyên).

Độ phức tạp: $O(m^2 \log(n))$.

L

Lời giải: Lê Anh Đức (amazingbamboo_with_coccoc)

Gọi X, Y là độ dời của vị trí sau N bước so với X_s, Y_s . Khi đó X, Y là các biến ngẫu nhiên xác định trong đoạn $[-N, N]$

Đề bài yêu cầu tính:

$$answer = E((X_s + X)^2 + (Y_s + Y)^2)$$

Sử dụng tính chất [linearity of expectation](#), ta có:

$$answer = E((X_s + X)^2) + E((Y_s + Y)^2)$$

Nhận thấy tính độc lập của các chiều x và y , ta tiếp tục với chiều x , chiều y hoàn toàn biến đổi tương tự:

$$\begin{aligned} E((X_s + X)^2) &= E(X_s^2 + 2 * X_s * X + X^2) \\ &= E(X_s^2) + E(2 * X_s * X) + E(X^2) \\ &= X_s^2 + 2 * X_s * E(X) + E(X^2) \end{aligned}$$

Đến đây chỉ cần tính được $E(X)$ và $E(X^2)$ là bài toán giải xong, chú ý là đề bài yêu cầu in ra giá trị chính xác của kỳ vọng nên ta không thể tính toán xấp xỉ.

Đi tính các giá trị kỳ vọng trên đối với biến X chính là đi giải bài toán một chiều: "Trên trục số bắt đầu từ điểm 0 ta di chuyển N bước; tại mỗi bước, sang trái với xác suất

$p = c/(a + b + c + d)$, sang phải với xác suất $q = a/(a + b + c + d)$, giữ nguyên vị trí với xác suất $1 - p - q$ ".

Như vậy X là tổng của N biến ngẫu nhiên độc lập x_i cùng phân phối:

- $P(x_i = -1) = p$
- $P(x_i = +1) = q$
- $P(x_i = 0) = 1 - p - q$

Ta có:

- $E(X) = \sum_{i=1}^N E(x_i) = N * E(x_i) = N * (q - p)$
- $E(X^2) = E((\sum_{i=1}^N x_i)^2)$
 $= E(\sum_{i=1}^N x_i^2) + E(\sum_{i=1}^N \sum_{j=1, j \neq i}^N x_i * x_j)$

Lại có:

$$E(x_i^2) = p * (-1)^2 + q * (+1)^2 = p + q$$

$$E(x_i * x_j) = p * p * (-1) * (-1) + q * q * (+1) * (+1) + p * q * (-1) * (+1) + q * p * (+1) * (-1) = (p - q)^2$$

Nên:

$$E(X^2) = N * (p + q) + (N^2 - N) * (p - q)^2$$

Vậy:

$$E((X_s + X)^2) = X_s^2 + 2 * X_s * N * (q - p) + N * (p + q) + (N^2 - N) * (p - q)^2$$

Đề bài yêu cầu tính kết quả theo modulo $10^9 + 7$, nên các số p và q cũng cần biểu diễn theo modulo. Chú ý là các xác suất đều là số hữu tỉ, và phép chia có thể tính bằng cách nhân với [nghịch đảo modulo](#).