SOLUTION – KIỂM TRA LẦN 1

GỢI Ý VỀ THUẬT TOÁN

Câu 1*: (6,0 điểm)* **Mã định danh.**

Subtask 1: $M \le 10; n \le 3; k \le 2 \times 10^5$

- Nếu $n = 1$, ta có: $f(1,k) = kf(0,k) = k$;
- Nếu $n = 2$, ta có: $f(2,k) = \frac{k(f(0,k)+f(1,k))}{2} = \frac{k(1+k)}{2}$;
- Nếu $n = 3$, ta có: $f(3,k) = \frac{k(f(0,k)+f(1,k)+f(2,k))}{3} = \frac{k(1+k+\frac{k(k+1)}{2})}{3} = \frac{k(1+k)(2+k)}{6}$.

Căn cứ vào các công thức trên, tính kết quả của hàm lấy theo modulo $10^9 + 7$. Độ phức tạp: $O(M)$.

Subtask 2: $M \le 2 \times 10^5;\ n + k \le 65$

Với $n \ge 1$, ta có:

$$f(n,k) = \frac{k(f(0,k)+f(1,k)+\cdots+f(n-2,k)+f(n-1,k))}{n} = \frac{n-1}{n} \times \frac{k(f(0,k)+f(1,k)+\cdots+f(n-2,k)]}{n-1} + \frac{kf(n-1,k)}{n}$$

$$= \frac{n-1}{n} f(n-1,k) + \frac{k}{n} f(n-1,k) = \frac{(n-1+k)f(n-1,k)}{n}.$$

Từ công thức hàm truy hồi ta xây dựng hàm đệ quy để tính giá trị của hàm và lấy kết quả theo modulo $10^9 + 7$. Với hàm đệ quy, kết quả có thể bị tràn số khi $n + k > 65$. Độ phức tạp: $O(Mn)$.

Subtask 3: $M \le 5 \times 10^3;\ n, k \le 5 \times 10^3;$

Từ công thức hàm truy hồi trên, ta có:

$$f(n,k) = \frac{(n-1+k)}{n} \times f(n-1,k) = \frac{(n-1+k)}{n} \times \frac{(n-2+k)}{n-1} \times f(n-2,k) = \cdots$$

$$f(n,k) = \frac{(n-1+k)(n-2+k)\ldots(1+k)k}{n!}$$

Để tính $f(n,k)$ % $(10^9 + 7)$:

1. Tính các giá trị:

   $a = (n-1+k)(n-2+k)\ldots(1+k)k$ % $(10^9 + 7)$.

   $b = n!$ % $(10^9 + 7)$

2. Dùng định lí Ferma nhỏ tính c là nghịch đảo mô đun $p = 10^9 + 7$ (nguyên tố) của $n!$: $c = b^{p-2}$ % p.
3. Kết quả của $f(n,k)$ % p là: $(a \times c)$ % p. Độ phức tạp: $O(Mn)$.

Subtask 4: $1 \le M \le 2 \times 10^5;\ 1 \le n, k \le 2 \times 10^5$

Ta có: $f(n,k) = \frac{(n-1+k)(n-2+k)\ldots(1+k)k}{n!} = \frac{(n-1+k)!}{n!(k-1)!}$

Để giảm độ phức tạp ta tính trước hai mảng sau:

$f[i] = i!$ % p,

$rf[i] = (f[i])^{p-2} \% p$

với $p = 10^9 + 7, i = 0,1, \ldots, 2 \times (2 \times 10^5)$.

Khi đó: $f(n,k) \% p = (f[n-1+k] \times rf[n] \% p) \times rf[k-1] \% p$. Độ phức tạp: $O(M.(n+k))$.

Câu 2:*(7,0 điểm)* **Trồng hoa.**

Subtask 1:

Với $N, M \leq 100$ dùng mảng đánh dấu các ô thỏa mãn, sau đó dùng *prefix sum* và duyệt hết tất cả các cặp ô (xs, ys) và (xe, ye) và kiểm tra xem có thỏa mãn điều kiện không với độ phức tạp $O(N * M)^2$.

Subtask 2:

Với $N, M \leq 4000$ Bài toán tính số hình chữ nhật con mà tất cả các ô của nó đều được tưới nước và tưới phân vi sinh, đầu tiên sử dụng mảng đánh dấu để dánh dấu các ô thỏa mãn, sau đó gọi $H(i,j)$ là số ô liên tiếp bắt đầu từ ô (i,j) lên phía trên. Gọi $Bot,\ Top,$ $Left, Right$ lần lượt là biên dưới, trên, trái, phải của hình chữ nhật con ta xét.

Duyệt hết tất cả các *Bot*, với mỗi *Bot* ta duyệt *Top* từ trên xuống dưới và dùng cấu trúc dữ liệu **DSU** để đếm xem có bao nhiêu cặp *(Left, Right)* thỏa mãn. Độ phức tạp $O(N)^2$.

Subtask 3:

Với $N \leq 4000$ ta sẽ dễ dàng thấy nếu có 2 máy phun nước đặt ở 2 ô $(x_1, y_1), (x_2, y_2)$ và $x_1 \leq x_2, y_1 \leq y_2$ thì máy phun nước ở ô (x_2, y_2) sẽ không quan trọng nữa, điều này cũng tương tự với máy tưới phân. Bởi vậy ta sẽ xây dựng hàm W_i với $W_i = k$ có nghĩa k là số bé nhất thỏa mãn tồn tại 1 máy phun nước ở ô (x,k) với $x \leq i$, Hàm F_ivới $F_i = k$ có nghĩa k là số lớn nhất thỏa mãn tồn tại 1 máy tưới phân ở ô (x,k) với $x \geq i$. Khi đó ta duyệt tất cả cặp *Bot, Top* và số cặp *(Left, Right)* thỏa mãn là $(F_{Bot} - W_{Top}) * (F_{Bot} - W_{Top} + 1)/2$ nếu $(W_{Top} > F_{Bot})$ thì không có ô thỏa mãn yêu cầu của bài toán. Độ phức tạp $O(N)^2$.

Subtask 4:

Với $N \leq 3 * 10^5$ sử dụng ý tưởng tương tự *subtask 3*, tuy nhiên ta không cần duyệt hết tất cả các cặp *(Bot, Top)*, ta sẽ duyệt tất cả các chỉ số *Bot* và dùng công thức để tính như sau:

$$(F_{Bot} - W_{Top}) * (F_{Bot} - W_{Top} + 1)\frac{1}{2}$$
$$= ({W_{Top}}^2 - 2 * W_{Top} * F_{Bot} - W_{Top} + {F_{Bot}}^2 + F_{Bot})/2.$$

Qua đây ta sẽ dùng công thức để tính tổng trên cho tất cả các *Top* sao cho $(W_{Top} \leq F_{Bot})$ sử dụng hai con trỏ để tính với độ phức tạp $O(N)$.

Subtask 5:

Với $N, M \leq 10^9$. Ta sử dụng ý tưởng của subtask 4, sẽ phân các hàng thành các blocks, với mỗi blocks sẽ bao gồm một tập các hàng liên tiếp nhau. Trước tiên với mỗi vòi

phun bất kể loại nào ở ô (x, y) thì ta thêm x và $x + 1$ vào danh sách hàng, sau đó ta *sort* danh sách hàng lại và lọc các phần tử trùng nhau. Gọi danh sách là Lis, khi đó Lis_icó block kéo dài từ Lis_i đến $Lis_i - 1$, gọi là $Length_i$. Áp dụng tư tưởng như ***Subtask 4*** tuy nhiên công thức sẽ là:
$= \frac{W_{Top}{}^2 - 2 * W_{Top} * F_{Bot} - W_{Top} + F_{Bot}{}^2 + F_{Bot}}{2} * Length_{Top} * Length_{Bot}$ nếu *Top* khác *Bot*, nếu *Top* = *Bot* và $(W_{Top} \leq F_{Bot})$ thì sẽ là $Length_{Top} * (Length_{Bot} + 1)/2$. Khi đó ta sẽ dùng 2 con trỏ và tách công thức ra để tính toán như subtask 4**.** Độ phức tạp $O(LIS.size() * q)$.

Câu 3:*(7,0 điểm)* **Du lịch.**

Bài toán phát biểu như sau:

Cho đồ thị $G = (V, E)$ có n đỉnh và m cạnh, đồ thị liên thông, có q truy vấn mỗi truy vấn gồm có hai số tương ứng với hai đỉnh u, v yêu cầu tìm số đỉnh lớn nhất trong hành trình $u -> v$ sao cho trên đường đi từ $u -> v$ đi qua các đỉnh $v' <= v$ và các cạnh có thể đi lại nhiều lần.

Subtask 1:

Với $q \leq 1000, s + 1 = t$ cần kiểm tra xem s có đến được t thông qua các đỉnh có chỉ số $\leq t$ hay không. Việc này có thể thực hiện dễ dàng bằng *DFS/BFS* với độ phức tạp $O((n + m) * q)$.

Subtask 2:

Với $n \leq 1000, q \leq 1000$ tham lam bằng cách luôn đi đến một địa điểm nhỏ nhất có thể đi được cho đến khi tới t. Cài đặt tương tự như thuật toán *dijkstra* với độ phức tạp $O((n + m) * q * log(n))$.

Subtask 3:

Với $n <= 1000$ sắp xếp lại truy vấn, thực hiện cùng lúc các truy vấn có cùng s vì (s, t') có thể tính được trong quá trình tính (s, t) $(t' < t)$ với độ phức tạp $O(n * (n + m) * log(n))$.

Subtask 4:

Với giới hạn của bài toán sử dụng **DSU** để giải: ta thấy rằng bản chất của mỗi truy vấn (s, t) là đếm xem có bao nhiêu $(s <= t' <= t)$ sao cho đi được từ s đến t' mà chỉ dùng các đỉnh có chỉ số $<= t'$. Điều này hợp lý vì dãy các điểm chụp ảnh là dãy tăng nên với mọi điểm có thể chụp ảnh ta luôn chụp được mà không cần quan tâm tới thứ tự di chuyển.

Duy trì **DSU** chứa k đỉnh đầu tiên, ta có thể trả lời nhanh cho tất cả cặp (s, t') với $s <= k$ và $t' = k$. Việc còn lại là sử dụng mảng dồn để tính kết quả, hoặc xử lý truy vấn theo thứ tự tăng dần t, với độ phức tạp $O(\max(n * (n + m) * log(n)), q)$.

--------HẾT--------