Bài 1: Xâu đại (BSTR)

Subtask 1: $n \leq 20$

Ta sẽ duyệt nhị phân để tìm ra được xâu có thứ tự từ điển lớn nhất với từng độ dài k và lưu lại.

Độ phức tạp: $O(2^n + q)$ hoặc $O(20 * 2^n + q)$.

Subtask 2: $n \leq 200$

Ta sẽ sử dụng thuật toán tham lam

Gọi xâu T là xâu đại độ dài k. Xét kí tự thứ p của xâu T, ta cần tìm kí tự thứ j của xâu S sao cho $i < j$ với i là vị trí của $T[p-1]$ trên xâu S, $S[j]$ là kí tự max trên đoạn từ $i+1 \rightarrow n-p+1$.

Với mỗi xâu đại độ dài k, ta lưu lại xâu và trả lời.

Độ phức tạp: $O(n^3 + q)$.

Subtask 3: $n \leq 2000$

Nhận xét: Ta có thể xây dựng xâu đại độ dài k dựa vào xâu đại độ dài $k-1$

Để xử lí được subtask này, ta sẽ sử dụng stack.

Gọi $last$ là vị trí cuối cùng trên xâu S mà chưa được thêm vào, ban đầu $last = n$. Gọi pos là kí tự cuối cùng được thêm vào của xâu đại độ dài $k-1$, cũng là phần tử cuối cùng được thêm vào của stack, ta dễ dàng thấy:

- TH1: Nếu $pos < last$, chúng ta vẫn có thể tìm max từ $pos + 1 \rightarrow last$ trên xâu S và thêm vào xâu kết quả, đồng thời thêm vào stack.
- TH2: Ta sẽ xét đến những điểm trước pos, ta sẽ pop phần tử cuối cùng và gán lại $last$ bằng phần tử đó

Lặp lại cho tới khi vào TH1, nghĩa là lấy được 1 vị trí mới.

Để lấy được max trong một đoạn nhanh, ta có thể dùng RMQ hoặc Segment Tree

Độ phức tạp: $O(q + 2n^2 logn)$.

Subtask 4: $n \leq 10^5$

Ta sẽ không cần cộng thêm kí tự vào xâu nữa mà sẽ dùng một Segment Tree để update những vị trí được thêm vào, với mỗi câu hỏi ta sẽ walk on tree để trả lời.

Độ phức tạp: $O(q + 2nlogn)$.

Code: https://ideone.com/hwpJs3

Bài 2: Hai chú ếch (FROG)

Subtask 1: $n \leq 200$

Ta sẽ $dp(i,j,p)$ nghĩa là con ếch thứ nhất đang ở vị trí i, con ếch thứ hai đang ở vị trí j và đang đến lượt con ếch thứ p nhảy. Mỗi lần chuyển nhãn, ta sẽ for vị trí k sao cho chênh lệch vị trí mới của một con ếch với vị trí con ếch còn lại không quá d.

$dp(i,j,0) = dp(k,j,0/1) + a[0][k]$

$dp(i,j,1) = dp(i,k,1/0) + a[1][k]$

Độ phức tạp: $O(2n^3)$.

Subtask 2: $n \leq 2000$

Ta sẽ không cần mất $O(n)$ để chuyển nhãn nữa, với mỗi lần chuyển lượt ếch ta bắt buộc phải cộng giá trị ở vị trí đó, còn những vị trí con ếch đi qua ta có thể lựa chọn cộng hoặc không.

Độ phức tạp: $O(2n^2)$.

Subtask 3: $n \leq 10^5$

Nhận xét: Con ếch có thể nhảy đến tất cả các ô có giá trị dương, vì vậy ta sẽ có mảng g được tạo như sau:

$a[p][i] \geq 0 \rightarrow g[p][i] = 0, a[p][i] < 0 \rightarrow g[p][i] = -a[p][i]$

Vậy ta sẽ chuyển về bài toán sau: Tìm cách nhảy của hai con ếch sao cho giá trị đạt được là nhỏ nhất.

Kết quả của bài toán ban đầu là: Tổng giá trị dương – kết quả bài toán con.

Để giải quyết bài toán con ta sẽ làm như sau:

Gọi $dp(p,i)$ là tới lượt của con ếch p và nó đang ở vị trí i. Ta có 2TH:

TH1: Chỉ có con ếch p nhảy tới i, lúc đó ta cần tìm $min(dp(1-p,j)$ với $i-d \leq j \leq i$) + $g[p][i]$ (nhảy luân phiên)

TH2: Để con ếch p nhảy tới i, con ếch còn lại phải nhảy tới j trước, vậy kết quả sẽ là $min(dp(p,i')$ với $i-d \leq i' \leq i$) + min($g[1-p][j]$ với $i-d \leq j \leq i$) + $g[p][i]$ (con ếch còn lại phải nhảy thêm để con ếch p có thể đi với giá trị nhỏ hơn)

Để lấy min nhanh ta có thể dùng Segment Tree hoặc Deque Min-Max để xử lí

Độ phức tạp: $O(n)$ hoặc $O(nlogn)$.

Code: https://ideone.com/ssvdiO

Bài 3: Robot tô màu (Robot)

Subtask 1: $n \leq 10$

Ta xét lần lượt các hoán vị và duyệt nhị phân để chọn, nhưng như vậy thì độ phức tạp sẽ rất lớn: $O(n! * 2^n)$

Vì vậy ta đơn giản chỉ lấy k máy đầu tiên của hoán vị đó và kiểm tra xem có thỏa mãn hay không

Độ phức tạp: $O(n * n!)$

Subtask 2, 3, 4: $n, q \leq 10^5$

Nhận xét rằng chúng ta có thể chuyển bài toán qua mô hình đồ thị.

Mỗi một máy là một cạnh nối từ đỉnh a sang đỉnh b Nhiệm vụ của chúng ta là liên thông tất cả các đỉnh, đồng thời in ra theo thứ tự Để tính toán được chi phí, ta nhận ra đây là bài toán tìm cây khung nhỏ nhất.

Để xử lí việc in ra theo thứ tự, ta sẽ in ra các cạnh theo thứ tự dfs lấy 1 làm gốc.

Độ phức tạp: $O(nlogn + mlogm)$

Code: https://ideone.com/8flgTy

Bài 4: Hoán vị (PER)

Subtask 1: $n, q \leq 1000$

Với mỗi truy vấn, ta có thể for từ $x \to y, u \to v$ để đánh đấu những số đã xuất hiện trong cả hai dãy và in ra kết quả.

Độ phức tạp : $O(n^2)$

Subtask 2: $x = u$ và $y = v$ với mọi câu hỏi

Subtask này khiến ta dễ dàng nhận ra đây là thuật toán MO.

Độ phức tạp: $O(n\sqrt{n} + q\sqrt{n})$

Subtask 3: $x = 1$ với mọi câu hỏi

Gọi p_i là vị trí của số b_i trong dãy a. Bài toán trở thành đếm số vị trí $x \leq i \leq y$ mà $u \leq p_i \leq v$

Với subtask này, ta có thể sắp xếp các truy vấn theo y, và từ đó tính toán kết quả truy vấn sau dựa trên kết quả truy vấn trước.

Do ở các truy vấn, y được sắp xếp tăng dần, nên ta có thể duy trì một mảng đếm c_i là số giá trị trong dãy $(p_1, p_2, \dots, p_y)$ có giá trị bằng i. Với mỗi truy vấn, ta cần một cấu trúc dữ liệu xử lí được hai yêu cầu:

- Tăng giá trị ở vị trí bất kỳ lên 1.
- Tính tổng các giá trị trong đoạn (u, v).

Ta có thể sử dụng Segment Tree để xử lí với độ phức tạp $O(nlogn)$

Subtask 4: $n, q \leq 10^5$

Hướng tiếp cận 1: xử lí offline, Segment Tree

Gọi $f(x, y, u, v)$ là số lượng số trong đoạn $p_x, p_{x+1}, \dots, p_y$ có giá trị trong khoảng u đến v. Dễ thấy $f(x, y, u, v) = f(1, y, u, v) - f(1, x-1, u, v)$.

Đến đây ta có thể chuyển về subtask 3 để xử lý.

Độ phức tạp: $O(nlogn)$

Code: https://ideone.com/0turzl

Hướng tiếp cận 2: xử lí offline, chia căn

Ta quay trở về yêu cầu của subtask 1: Đếm số phần tử trong đoạn $p_x, p_{x+1}, \dots, p_y$ có giá trị trong đoạn (u, v).

Ta có thể chia dãy thành $\sqrt{n}$ block, với mỗi block ta có thể xử lý trong $O(1)$ bằng cách đếm các số trong block và tạo một mảng tiền tố.

Gọi $cnt[i][t]$ là số lượng số trong block i có giá trị bằng t, và $pre[i][t] = cnt[i][1] + cnt[i][2] + \dots + cnt[i][t]$, ta có thể đếm số lượng số trong block i có giá trị trong đoạn (u, v) bằng cách lấy $pre[i][v] - pre[i][u-1]$.

Độ phức tạp: $O(n\sqrt{n})$

Bài 5: Hang động (CAVE)

Tóm tắt đề

Cho một cây n đỉnh, mỗi đỉnh $1 \leq i \leq n$ có mức độ yêu thích p_i và một số $c_i = \{0, 1, 2, 3\}$.
Gọi $G(u, v) = \sum p_k$ với k thuộc đường đi từ u đến v.
Tìm $max(G(i, j))$ mà $c(i) \mid c(j) = 3$ và $(i), c(j) > 0$ $(*)$.

Subtask 1: Dạng line $n \leq 2000$

Với $u_k = k, v_k = k - 1$ thì cây đã cho là cây suy biến thành đường thẳng. Ta chỉ cần duyệt qua các cặp i, j và cộng tổng p_k $(i \leq k \leq j)$ vào đáp án bài toán nếu $(*)$ thỏa mãn.

Subtask 2: Dạng tree $n \leq 2000$

Với $n \leq 2000$, ta lần lượt for qua các đỉnh u từ 1 đến n, rồi duyệt qua tất cả các đỉnh v thuộc cây con gốc u, cập nhật đáp án tương tự như subtask 1.
Độ phức tạp: $O(n^2)$.

Subtask 3: Dạng line $n \leq 10^5$

Nhận xét: Với mỗi j và v = {1,2,3} chỉ tồn tại tối đa một vị trí i < j và c_i = v tối ưu mà ta có thể cập nhật cho đáp án bài toán.

Gọi $pref(i) = \sum p_j$ với $j < i$. Lúc này $G(i, j) = pref(j) - pref(i - 1)$. Ta duyệt j tăng dần rồi tìm $pref(i)$ nhỏ nhất với $i < j$ thỏa mãn $(*)$. Để tối ưu thì trong lúc duyệt, ta lưu lại mảng $opt(v) = min(pref(i))$ mà $i < j$ và $c_{i+1} = v$. Nhờ đó ta có thể tìm ngay tổng tối ưu.

Độ phức tạp: $O(3n)$ hoặc $O(n)$.

Subtask 4: Dạng tree $n \leq 10^5$

Với mỗi đỉnh u, ta xét hai loại đường đi tạo ra $G(u, v)$ lớn nhất
1. Đường đi từ u đến một đỉnh v thuộc cây con gốc u
2. Đường đi từ v_1 đến v_2 là hai đỉnh thuộc hai nhánh khác nhau trong cây con gốc u.

Đến đây thì có khá nhiều cách triển khai, tác giả xin trình bày cách làm như sau

Ta dfs từ một đỉnh bất kì, với mỗi đỉnh u ta lưu $ma[u][x]$ là $G(u, v)$ lớn nhất mà v thuộc subtree của u và $c_v = x$ (dfs đến các con của u trước khi cập nhật kết quả). Từ đó ta cập nhật được luôn đáp án loại 1. Đối với loại 2, ta lần lượt for qua v là các con trực tiếp của u rồi vừa tìm $G(v, v')$ tối ưu vừa cập nhật $ma[u][1 \ldots 3]$.
Code: https://ideone.com/1Sd6bb

Bài 6: Biến đổi bảng số (CTAB)

Subtask 1, 2: $n \leq 3$

Coi mỗi một trạng thái của bảng là một đỉnh của đồ thị. Phép biến đổi để chuyển từ trạng thái này qua trạng thái khác tương ứng cạnh đồ thị. Áp dụng thuật toán BFS/DFS để tìm đường đi tương ứng với tìm cách biến đổi từ trạng thái xuất phát tới trạng thái đích.

Subtask 3: **$n = 4$ và tồn tại cách biến đổi không quá 25 bước**

Loang từ hai đầu, mỗi đầu loang 12 bước, không tìm được ghi 25.

Subtask 4, 5: $n \leq 5$

Dùng thuật toán A* hoặc thuật toán Heuristic với thông tin định hướng tới đích là số vị trí sai khác của bảng hiện tại với bảng đích.

Code: https://ideone.com/atfVYv

Solution được thực hiện bởi nhóm TLEOJ