SOLUTION DAY 10/10

Bài 1: Giá trị lớn nhất

n: độ dài của a.
Subtask 1: Duyệt $2n \times 2^n$

Subtask 2:
Đầu tiên ra đảo ngược 2 dãy số a và b để thuận tiện cho việc truy vết sau này.
Nhận xét: c lớn nhất đồng nghĩa với độ dài của c là lớn nhất.
Quy hoạch động $f[i][j]$ là độ dài độ dài dãy con chung dài nhất khi xét i phần tử đầu tiên của dãy a và j phần tử đầu tiên của dãy b.
bit1[i]: lưu vị trí gần nhất của phần từ có giá trị là i trong dãy số a
bit2[i]: lưu vị trí gần nhất của phần từ có giá trị là i trong dãy số b

- Nếu $a[i] = b[j]$:
$f[i][j] = f[i-1][j-1] + 1$
$bit1[a[i]].update\ (i, i);$
$bit2[a[i]].update\ (j, j);$

- Nếu $a[i] \neq b[j]$:
$f[i][j] = max(f[i-1][j], f[i][j-1])$

Truy vết:
mx: độ dài của xâu con chung dài nhất a và b.
Duyệt từ $cnt = mx$ cho đến khi $cnt = 0$
- Bắt đầu duyệt từ $i = n, j = n$.
- Duyệt num từ 9 về 0:
    + Tìm x và y là vị trí gần i nhất mà $a[x] = num, b[y] = num$.
    + Nếu f[x][y] = mx ta sẽ lưu lại con chung a[x] và đặt i = x-1, j = y-1, --cnt;

Kết quả là dãy con đã lưu.

Bài 2: Quản lý lương

Đầu tiên, ta cần tạo một cây giữa các nhân viên. Qua một lần DFS từ nhân viên thứ nhất, ta có thể tìm được thứ tự vào/ra và thứ tự Topo của từng nhân viên. Gọi mảng thứ tự đó lần lượt là $in, out, topo$.

Với mỗi một truy vấn cộng $p\ A\ x$, ta sẽ cộng từng thành phần trong cây con gốc A một khoảng là x. Ta nhận thấy rằng, khi đó ta đang cộng một đoạn liên tiếp các giá trị trong

mảng $topo$, bắt đầu từ phần tử thứ $in[A] + 1$ tới phần tử $out[A]$. Như vậy, ta có thể dễ dàng sử dụng Segment Tree để cập nhật tổng trên một khoảng.

Với mỗi truy vấn u A, ta sẽ cần in ra giá trị của phần tử thứ $in[A]$.

Suy ra, ta cần xây Segment Tree dựa trên mảng a ban đầu, sau đó thực hiện hàm lấy giá trị và hàm cập nhật tương tự như bài Cập nhật khoảng Truy vấn điểm.

Độ phức tạp thuật toán là $O(m \times \log_2 n)$.

Bài 3: Truy bắt tội phạm

Mấu chốt của bài toán này là phải tìm được 2 đỉnh xuất phát và kết thúc của k đỉnh đã cho, từ đó ta có thể suy ra được các đỉnh thỏa mãn sẽ là từ một trong 2 đỉnh tìm được đi qua k đỉnh đã cho và đi đến các đỉnh này.

Subtask 1: Đồ thị cho là dạng đường thẳng, do đó ta chỉ cần tìm 2 đỉnh lá sau đó tiến dần vào trong để tìm hai đỉnh xuất phát và kết thúc trong k đỉnh đã cho, kết quả là những đỉnh từ 2 đỉnh tìm được đi đến 2 lá tương ứng.

Subtask 2: Đồ thị dạng cây, ta coi một trong k đỉnh đã là cây có gốc và dfs từ gốc xuống, khi dfs đến đỉnh u ta sẽ đếm xem nó có bao nhiêu nhánh chứa các đỉnh trong k đỉnh đã cho gọi là c_u, nếu u không phải là gốc thì $c_u \leq 1$ thì mới tồn tại đường đi nhỏ nhất qua k đỉnh đã cho, ngược lại thì vô nghiệm. Nếu u là 1 đỉnh trong k đỉnh đã cho và c_u = 0 có nghĩa u chính là 1 trong 2 đỉnh xuất phát và kết thúc. Nếu ta tìm đc 2 đỉnh có c_u = 0 thì đấy chính là 2 đỉnh cần tìm, ngược lại thì đỉnh gốc chính là đỉnh còn lại. Từ 2 đỉnh tìm được ta loang ra các đỉnh có c_u = 0, kết quả chính là số lượng đỉnh tìm được khi loang ra.

Subtask 3, 4: Duyệt từng đỉnh y, ta có $d[y][u]$ là khoảng cách nhỏ nhất từ y đến u. Dựa vào $d[y][u]$ ta có thể biết thứ tự cần đi qua k đỉnh đã cho như thế nào (đỉnh u đi qua trước v trên đường đi từ x đến y nếu $d[y][u] > d[y][v]$). Sau khi có thứ tự $u_1, u_2, \dots, u_k$ ta có thể kiểm tra được y có phải đỉnh thỏa mãn hay không dựa vào điều kiện $d[y][u_i] = d[u_i][u_{i+1}] + d[y][u_{i+1}]$. Với $n \leq 100$ ta có thể sử dụng luôn floyd để xây dựng mảng d, với $n \leq 1000$ thì cần dùng Dijkstra và hàng đợi ưu tiên.

Subtask 5: Với subtask này ta tìm được 2 đỉnh xuất phát và kết thúc trong 2 lần Dijkstra, lần đầu là Dijkstra từ 1 đỉnh bất kì trong tập k đỉnh và tìm đỉnh xa nhất s, và Dijkstra lần 2 để tìm đỉnh t còn lại. Sau khi tìm được 2 đỉnh này, ta Dijkstra trạng thái từ 2 đỉnh này đến các đỉnh trong đồ thị với mỗi trạng thái là $(u, state)$ với $state$ ở đây mô tả những đỉnh trong tập k đỉnh đã đi qua. Những đỉnh thỏa mãn sẽ có $d[u][state] = ds[u]$

($d[u][state]$ là đường đi ngắn nhất đến u và trạng thái qua các đỉnh trong tập k đỉnh là $state = 2^k - 1$, $ds[u]$ là đường đi ngắn nhất từ s đến u).

Độ phức tạp sẽ là $O(n * 2^5 * (logn + 5))$.

Subtask 6: Giống subtask 5 ở tìm s và t, tuy nhiên ta sẽ quy hoạch động $c[u]$ là số đỉnh nhiều nhất trong tập k đỉnh đã cho trong các đường đi ngắn nhất từ s đến u. Như vậy những đỉnh u thỏa mãn sẽ có $c[u] = k$. Độ phức tạp chỉ là $O((n + m)\ log\ n)$.

HẾT