**TRAIN**

Ta sẽ xây “cây” BFS với gốc là đỉnh 1 ( thủ đô).

Lưu ý là trong cây này, nếu từ u đến được v, v đã thăm rồi và v không phải cha u trong cây, thì vẫn có cạnh nối từ u -> v.

Với mỗi đỉnh, ta dùng 2 set lưu lại danh sách các đỉnh kề với nó trên cây ( 1 cho các thằng tiền bối, 1 cho bọn hậu bối).

1 đỉnh chỉ còn tốt khi mà nó còn đường đến 1.

Ta thực hiện lần lượt từng thao tác.

Khi xóa 1 cạnh, gọi u và v là 2 mút ( u là cha của v).

Xóa v trong set dưới của u, xóa u trong set trên của v.

Kiểm tra coi set trên của v có rỗng không, nếu rỗng

Có nghĩa là v là 1 đỉnh chết ( ans++), sau đó lần lượt xóa các cạnh từ v -> set dưới của v, và làm tương tự.

Xóa mỗi cạnh max 1 lần nên đpt là O (MlogM).

**ESCAPE**

Bài này chỉ là Dijkstra bình thường.

**Count**

H ta thấy rằng, do a[i] – I <= k -> tại vị trí I chỉ chọn được các số từ max(1, I – k) -> min(n, I + k) ( gọi là khoảng XXX).

Kết hợp với k nhỏ, ta nghĩ đến dp bitmask

Dp[i][mask ] = số hoán vị đã xây đúng, mà đang xét đến số I, các số đã được chọn trong khoảng XXX thể hiện qua mask.

Đến đoạn này thì còn tùy cách code.

Tham khảo code tại:

<https://pastebin.com/NCsnLdS2>

**Count2**

Bài này có một tính chất cực kỳ vi diệu.

**Dãy 1, 2, 3, 4, … n là 1 dãy tăng !**

Nhìn thì có vẻ ngu ngu, cơ mà đây là tính chất quyết định cách làm bài toán này :v. ( Cơ bản là hàm bài này là hàm swap nên ta nghĩ đến sort thôi, cụ thể hơn là bubble sort.

Ta sẽ có 1 thuật toán dp ( hiển nhiên rồi) trong O ( N ^ 3).

Dp[i][j] = số dãy đúng của B từ I -> j mà thực hiện xong câu lệnh trong cái đoạn đó thì dãy ai -> aj tăng.

Base: Dp[i][i] = 1;

Tính Dp[u][v].

Giả sử I, I + 1 là bước chuyển cuối cùng ( I = u .. v – 1).

=>Dp[u][v] = Dp[u][i] \* Dp[I + 1][v] \* X \* canMove(u, v, i)

Trong đó X = (v – u + 1) C (I – u).

Lý do có X là vì , dp[u][i] là số hoán vị đúng từ u -> I, dp[I + 1][v] là số hoán vị đúng từ I + 1 -> v, 2 thằng nầy độc lập với nhau => có thể xếp xen kẽ nhau => phải chọn thêm vị trí cho thằng đầu, có v – u + 1 vị trí, thằng đầu có I – u số -> X = Chỉnh hợp của cái trên

Còn cái canMove là, ta có dãy a[u] -> a[I] sorted, a[I + 1] -> a[v] sorted, thì giờ ta chuyển số a[i] và a[I + 1] thì cả dãy a[u] -> a[v] nó có sorted nữa ko, điều này đúng khi nào ?

Khi mà a[I + 1] > số max nhì của đám từ a[u] -> a[i], và a[i] < số min nhì của đám a[I + 1] -> a[v].

Cái này thì lập mảng tính trước khá dễ.

Bài toán đến đầy là xong.