BONUS

Bài này là một bài mảng cộng dồn cơ bản.

Với mảng cộng dồn, có thể tính tổng của một hình chữ nhật trên mảng 2 chiều trong đpt O(1).

Các bạn có thể đọc tài liệu về mảng cộng dồn 1 chiều và 2 chiều tại đây:

https://vnoi.info/wiki/algo/data-structures/data-structures-overview.md

Sau khi tính được mảng cộng dồn thì tổng của hình chữ nhật có góc trái trên và phải dưới là (i, j) và (u, v) là:

$$Sum = f[u][v] - f[i-1][v] - f[u][j-1] + f[i-1][j-1]$$

Cách giải thích cũng tương tự như cách các bạn đọc ở link trên.

Ở bài bonus có thể for qua các hình chữ nhật k x k với đpt O(n * n) và mỗi lần tính tổng mất O(1). => đpt = O(n * n * 1).

DRONE.

Bài này là một bài Quy hoach động

Nếu chưa làm quen với qhd, các bạn có thể xem một số bài qhd cơ bản tại đây:

(rồi bữa sau quay lại làm bài này =))))

https://vnoi.info/wiki/algo/dp/basic-problems.md

https://atcoder.jp/contests/dp/tasks

Ngoài ra trên vnoi còn có một số tài liệu về qhđ và rất nhiều bài tập từ dễ đến khó.

với bài DRONE:

Gọi f[i][x] là số hàng lớn nhất lấy được khi đang dừng tại điểm i và đã dừng x lần

Ta có thể tính f[i][x] = max(f[j][x - 1] + số_hàng[i])

Với j < i và từ ô j có thể nhảy tới ô i (a[j] + j >= i)

Có O(n * K) trạng thái, mỗi lần chuyển trạng thái phải duyện j từ 1 -> i => chuyển trạng thái mất O(n).

Vậy đpt là O(n * K * n). (có thể làm đc subtask 1)

Subtask còn lại: có thể thấy mức năng lực nạp được ở mỗi điểm <= 50 (a[j] <= 50)

Nên ta đưa ra nhận xét từ một điểm bất kì thì chỉ đi qua được tối đa 50 điểm khác về bên phải.

Hay từ mỗi điểm i thì chỉ có các điểm từ i – 50 đến i – 1 là đi được đến i vì a[j]<=50.

Nên với mỗi f[i][k] ta duyệt j từ i – 50 đến i – 1 mất O(50).

⇨ Đpt (n * K * 50) (đủ để AC).

VEXE

Đây là một bài kết hợp giữa thuật toán dijkstra (dùng để tìm đường đi ngắn nhất) và qhđ.

Các bạn có thể đọc tài liệu về dijkstra tại đây:

https://vnoi.info/wiki/algo/graph-theory/shortest-path.md

https://vnoi.info/wiki/algo/graph-theory/everything.md

(trên vnoi có rất nhiều bài tập từ cơ bản đến khó các bạn có thể luyện tập nhé)

Ở bài này:

Đặt d[u][x] là đường đi ngắn nhất từ đỉnh 1 tới đỉnh u và đã dùng x vé.

Giả sử d[u][x] đã được tính, với mỗi đỉnh v kề u (có cạnh u-v)

Ta cập nhật d[v] như sau

d[v][x] = d[u][x] + chi_phí[u – v] (Trường hợp không dùng thêm vé để đi qua cạnh u – v)

Nếu x < K (có nghĩa là vẫn có thể dùng thêm vé) ta có thể cập nhật d[v] như sau:

d[v][x + 1] = d[u][x] (dùng thêm 1 vé miễn phí để đi qua đỉnh v nên không mất chi phí)

Thuật toán dijkstra thông thường có đpt O(n * n).

Thông thường trong các kì thi thì dùng thuật toán dijkstra có cải tiến bằng cấu trúc dữ liệu heap có đpt O(n * log(n))

Đpt bài VEXE khi dùng dijkstra heap: O(n * log(n) * K).