Contest: <https://hnoj.edu.vn/contest/lopa2>

Pass: a22

1. Buy

* Dựng đồ thị với đỉnh 0 làm gốc
* Thêm cạnh từ 0 đến i (i từ 1 đến n), trọng số là p[i] (tượng trưng cho chi phí mua hàng)
* Với mỗi cạnh (ik,jk,c(ik,jk)) ta sẽ thêm cạnh từ ik đến jk, trọng số là c(ik,jk) \* 2 do ta phải đi quay lại (tượng trưng cho chi phí đi lại)
* Bài toán trở về tìm đường đi ngắn nhất từ 0 đến mọi đỉnh
* Code: https://ideone.com/47eVMU

1. Rescue

* Nhận xét rằng việc 2 bên đi từ đỉnh 1 và 1 đỉnh i và gặp nhau tại 1 điểm u tương đương với việc ta đi xuôi từ 1 đến u và đi ngược từ u đến i
* Dựng đồ thị: Ta sẽ dùng 2 danh sách liên kết để lưu các cạnh xuôi và ngược
* Sub 1,3 : a = b = 0

+ Lưu mảng d[u][t] là đường đi ngắn nhất từ đỉnh 1 đến đỉnh u, t = 0/1 là đến đỉnh u ta đi xuôi/ngược và dijkstra

+ Chuyển trạng thái: (i,0) -> (j,0) với trọng số là c(i,j) (cạnh xuôi) , cạnh (i,0) -> (j,1) với trọng số là c(j,i) (cạnh chuyển hướng, cạnh này đi ngược) , cạnh (i,1) -> (j,1) với trọng số là c(j,i) (cạnh ngược)

* Code sub 1 và 3: https://ideone.com/FC2g8w
* Full:

+ Lưu mảng d[u][t][sa][sb][cntb] là đường đi ngắn nhất từ đỉnh 1 đến đỉnh u, t = 0/1 là đến đỉnh u đang đi xuôi / ngược, sa là số lượng đường đi xuôi mà ta không tính chi phí, sb là số lượng đường đi ngược mà chi phí gấp đôi, cntb là số lượng đường đi ngược đã đi qua

+ Dijkstra từ đỉnh 1, chuyển trạng thái như trên, thêm các trường hợp bỏ chi phí cạnh (nếu t == 0) và gấp đôi chi phí cạnh (nếu t == 1). Ta thấy rằng nếu cntb <= b thì đáp án là d[u][t][0...a][cntb][cntb] còn nếu cntb >= b thì đáp án là d[u][t][b][cntb]. Vậy ta có thể gán cntb luôn <= b, và các giá trị trong mảng d thỏa mãn thì sb sẽ luôn = cntb

+ Giải thích kĩ hơn thì ta cần cntb là vì có trường hợp giá trị tối ưu trong mảng d sẽ là giá trị mà không gấp đôi cạnh nào (trong khi chúng ta cần phải gấp đôi đúng b cạnh). Còn ta không cần cnta vì giá trị tối ưu trong mảng d sẽ luôn là giá trị mà ta loại bỏ chi phí của nhiều cạnh nhất có thể.

* Code full: <https://ideone.com/nKoPNT>

1. Kth path

* Sub 1 (n,m <= 300) : dùng floyd:

+ For(k: 1 -> n)

For(i : 1 -> n)

For(j: 1 -> n)

If(c[i,j] > c[i,k] + c[k,j]) c[i,j] = c[i,k]+c[k,j]

+ Bản chất floyd là c[i,j,k]: độ dài đường đi ngắn nhất từ 1 đến j mà chỉ đi qua các cạnh từ 1 -> k

+ Có c[i,j,k] = max(c[i,j.k-1],c[i,k,k-1],c[k,j,k-1]

+ Có thể thấy phải for k trước để c[i,j,k-1] đã được tính xong. Trong cài đặt có thể bỏ chiều k đi.

* Full:

+ Dựng 1 đồ thị mới với k cạnh nhỏ nhất của đồ thị và các đỉnh thuộc các cạnh.

+ Có k <= 300 => đồ thị mới chỉ có tối đa 300 cạnh và 600 đỉnh (chú ý nén số hiệu đỉnh)

+ Floyd giống sub 1

* Code: <https://ideone.com/GmJuub>

1. DIJ

* Gọi C(i,j) là đường đi ngắn nhất từ i đến j, được đi qua các đỉnh từ 1 đến n. Tính trước mảng này bằng floyd như bên trên.
* Lưu ý input đề bài là 1 chiều, tức là chỉ update C(u,v) chứ không update C(v,u)
* Giả sử thêm 1 nút n+1. Giờ ta cần cập nhật mảng C khi được đi qua thêm đỉnh n+1
* Tính C(i,n+1):

For(i: 1 -> n)

For(j : 1 -> n)

C[i][n+1] = min(c[i][n+1],c[i][j],c[j][n+1])

* Tương tự ta tính C(n+1,i):

For(i: 1 -> n)

For(j : 1 -> n)

C[n+1][i] = min(c[n+1][i], c[n+1][j] + c[j][i])

* Sau đó ta tính lại C(i,j) khi thêm đỉnh n+1 với mọi i,j:

For(i: 1 -> n)

For(j: 1 -> n)

C[i][j] = min(c[i][j], c[i][n+1] + c[n+1][j])

* Khi thêm 2 nút n+2, n+3 ta làm tương tự. Khi đó ta sẽ thu được mảng C là độ dài đường đi của mọi cặp đỉnh (i,j) (1 <= i,j <= n+3)
* Code: <https://ideone.com/IdNq9O>

1. Stpath

* M = n-1 => đồ thị dạng cây
* M > n-1: Dựng cây khung với n-1 cạnh, và có thêm k = m – (n-1) cạnh nằm ngoài cây (k <= 21) (sử dụng DSU để dựng cây khung)
* Dijkstra trên k đầu mút (lấy đầu mút nào trong 2 đầu mút của cạnh đều được) của k cạnh (dijkstra 21 lần)
* Với mỗi cặp cạnh i,j có 2 TH:

+ TH1: đường đi từ i đến j chỉ được phép đi qua các cạnh trong cây khung => lca để tính đường đi, độ dài đường đi là d(root->i) + d(root->j) – 2 \* d(root->lca(i,j)) với root là gốc của cây khung mình chọn (trong code mẫu bên dưới root của cây khung là 1)

+ TH2: đường đi từ i đến j được phép đi qua mọi cạnh trong đồ thị . Nhận xét rằng chỉ có k cạnh nằm ngoài cây khung có thể ảnh hưởng đến đáp án => xét k đầu mút đã dijkstra, độ dài đường đi là d(u->i) + d(u->j) với u là 1 trong k đầu mút của k cạnh nằm ngoài cây khung

* Lấy min 2 trường hợp
* Code: https://ideone.com/B6KHs0