JOI 2024/2025 本選 3 問目
ミ・テレフェリコ (Mi Teleférico) 解説

解説: 蜂矢 倫久 (Mitsubachi)

Step 0

問題概要

0 問題概要

- ボリビアの首都ラパスにはミ・テレフェリコ (Mi Teleférico) というロープウェイが通っています
- 標高が 3700m 近くあり高低差のある移動は大変なので作られました
- 意味はスペイン語で「私のロープウェイ」

0 問題概要

- N 頂点 M 辺のグラフがあります

- 辺は有向辺で各辺には C_i という値がある
- 辺が A から B にあるとき、A < B が成立

- フリーパス (l, r) を用いると l ≦ C_i ≦ r なる辺を使える
- 使える辺のみで頂点 1 から全部の頂点に行けるようにしたい

- そこで次の操作を行う

0 問題概要

- フリーパス (l, r) とフリーパス (l', r') を交換する
- この操作のコストは $|l - l'| + |r - r'|$

- Q 個のクエリに答える j 番目のクエリは以下のように書ける
- フリーパス (L_j, R_j) を持っている状態からコスト X_j 以内の交換をして目標を達成できるか判定してください

0 制約

- $N, M \leqq 300\ 000$
- $Q \leqq 400\ 000$
- $P, X_j \leqq 10^9$
- $C_i, L_j, R_j \leqq P$

0 小課題

小課題番号	N, M	P	Q	X_j	配点
1	50		50	0	7 点
2		10			8 点
3		100			11 点
4		300 000		0	23 点
5		300 000			9 点
6	8 000				22 点
7					20 点

Subtask 1

$N, M, Q \leqq 50, X_j = 0$

1 $X_j = 0$ の意味

- $X_j = 0$ のとき可能な交換は $|l - l'| + |r - r'| = 0$ な場合だけ
- これは $l = l', r = r'$ なので異なるフリーパスと交換できない
- つまり「今あるフリーパスで全部いけるか判定してください」という形になる

1 グラフアルゴリズム

- フリーパスが決まってれば使える辺の一覧も $O(M)$ で分かる
- 使える辺が分かれば BFS, DFS, Dijkstra, ワーシャルフロイド などのアルゴリズムを用いてどの頂点に到達できるか分かる
- 計算量はクエリあたり BFS とかなら $O(N + M)$ で、ワーシャルフロイドなら $O(N^3 + M)$ で可能
- 全体として $O((N + M)Q)$ や $O((N^3 + M)Q)$ で解ける

1 発展的な話

- 時間に余裕がなければカットします

- ワーシャルフロイドでやってることは
- d[i][j] を i -> j の最短路として d[i][j] ← min(d[i][j], d[i][k] + d[k][j]) として求めると言うもの

- 正しいループ順:　k → i → j
- 間違ったループ順: i → j → k (などなど)

1 発展的な話

- 間違ったループ順でも 3 回やれば正しいことが知られてる
- 今回は i → j → k で 1 回しかしなくても正当

- 証明: i → j の最短路のうち k → j 辺を使うものがあるとして d[i][k] + d[k][j] を計算するとき d[i][k] が正しく計算されてれば OK

- j = i + 1, i + 2, ..., の順に帰納法を回すと OK

- よく見ると区間 DP の計算方法と同じ

Subtask 2

$P \leqq 10$

2 候補について考える

- フリーパスとしてありうるのは $1 \leqq l \leqq r \leqq P$ を満たすもののみ

- つまり $O(P^2)$ 種類しかない
- あるフリーパスが条件を満たすか → 小課題 1 で議論した通り BFS や DFS などで OK

- 前計算として $O((N + M)P^2)$ かけて全フリーパスについて条件を満たすか計算
- ワーシャルフロイドだと間に合わない

2 クエリの処理

- クエリの処理を考える
- 交換先は $O(P^2)$ 種類 コストはそれぞれ $O(1)$ で計算できる
- つまり交換する候補についてコストを計算して、交換可能ならそれが条件を満たすか前計算から確認すれば OK
- 全体で $O((N + M + Q)P^2)$ で解けました

Subtask 3

$P \leqq 100$

3 前計算について

- 小課題 2 と同様に前計算を考える
- 枝刈りできそうなパートを考える

- l を固定して r を動かすとき、OK になったら以降は見なくてよい (r を増やして使えた辺が使えなくなることはないため)
- l を増やして r を再び動かすとき、r のスタートは先ほど調べたときの r としてよい

3 具体例

- check(l, r) を (l, r) について条件を満たすか調べる関数とする

- 具体例で書いてみる P = 4 として
- l = 1: check(1, 1) → check(1, 2) → check(1, 3)
- l = 2: check(2, 3) → check(2, 4)
- l = 3: check(3, 4)
- l = 4: check(4, 4)

- という感じ

3 尺取り法

- 定数倍しか改善してなさそうだが実は check を呼び出す回数は $O(P^2)$ から $O(P)$ へ改善している
- いわゆる**尺取り法**というアイデア

- 証明: $l + r$ の総和を見ると、1 ずつ増えていく これと $l, r \leqq P$ より示される

- これを用いれば $O((N + M)P)$ で l を固定したときの r として必要な最小の値が分かる
- r として必要な最小の値を $f(l)$ とでも表記する

3 クエリの処理

- 各クエリでは交換するフリーパスの l を固定して考える

- 交換後の r について $f(l) \leqq r$ となるようにしたい
- これのために必要なコストは $\max(0, f(l) - R_j)$

- つまり $|L_j - l| + \max(0, f(l) - R_j) \leqq X_j$ なる l が存在するか探す
- これは l あたり $O(1)$ でできるのでクエリあたり $O(P)$

- よって $O((N + M + Q)P)$ で解くことができました

Subtask 4

$P \leqq 300\,000, X_j = 0$

4 条件の言い換え

- 今のところ使ってない問題の性質は $A_i < B_i$

- つまりグラフは DAG だということ
- DAG は有向辺からなる閉路がないグラフ

- これを使って条件が言い換えられるか考える

- とりあえず具体例で考えてみよう

4 条件の言い換え

- 頂点 2 に行きたいなら 1 → 2 の辺はほしい
- 頂点 3 に行きたいなら 1 → 3 か 2 → 3 はほしい
- 頂点 4 に行きたいなら 1 → 4 か 2 → 4 か 3 → 4 はほしい
- …

- とりあえず頂点 1 以外の各頂点について、その頂点に入る辺がほしい

- 実はこれが必要十分条件になっている
- 必要条件を列挙すると十分条件になることはよくある

4 証明

- 必要性: ある頂点 v について u → v という形の辺がないなら頂点 v にはどうやってもいけない
- 十分性: どの頂点からも辺を逆にたどって頂点 1 にいければ OK
- どの頂点からも辺は入ってきている この辺を逆にたどると番号の小さい頂点にいける
- これを繰り返せばいずれは頂点 1 にたどりつくので OK

4 証明

- 必要性: ある頂点 v について u → v という形の辺がないなら頂点 v にはどうやってもいけない
- 十分性: どの辺からも辺を逆にたどると頂点 1 にたどりつけばOK
- どの頂点からも辺は入ってきている この辺を逆にたどると番号の小さい頂点にたどりつく
- これを繰り返せばいずれは頂点 1 にたどりつくので OK

つまり、辺の情報として重要なのは B_i のみ

A_i を保存する必要はない！！

4 f(l) を求める

- 言い換えた条件をもとに f(l) を求めよう
- 条件は $l \leqq C_i \leqq r$ かつ $B_i = x$ である i が x によらずあること
- $B_i = x$ である i における C_i において l 以上の中で最小のものを使いたい
- 逆にこれを使えれば OK

4 具体例

- N = 3 において
- $B_i = 2$ である i における C の列が (3, 7, 9) で
- $B_i = 3$ である i における C の列が (4, 6, 9) とする

- l = 5 なら (3, 7, 9) からは 7 を、(4, 6, 9) からは 6 を使いたい
- これらの max である 7 が f(5)

- l = 10 なら (3, 7, 9) からも (4, 6, 9) からも取れない
- こういうときは f(10) = inf とでもしておこう

- 実装上は 10^{18} とか大きい数にしておけば OK

4 f(l) を求める

- f(l) を求める方法を考える

- B ごとに C を集めて sort した列を考える

- ある B について $(V_1, V_2, ..., V_n)$ とあるとき
- $i = 1, 2, ..., V_1$ について $f(i) \leftarrow \max(f(i), V_1)$ と更新
- $i = V_1 + 1, V_1 + 2, ..., V_2$ について $f(i) \leftarrow \max(f(i), V_2)$ と更新
- ...
- $i = V_n + 1, V_n + 2, ..., P$ について $f(i) \leftarrow \max(f(i), inf)$ と更新
- ということをしたい

4 f(l) を求める

- さっきの max が + の形ならいわゆる imos 法をすれば OK
- 累積 max を使うと今回も同様にできる
- $f(1) \leftarrow \max(f(1), V_1)$, $f(V_1 + 1) \leftarrow \max(f(V_1 + 1), V_2)$, ..., $f(V_n + 1) \leftarrow \max(f(V_1 + 1), \inf)$
- とあらかじめしておく 他の B についても同様
- 最後に $i = 2, 3, ..., P$ の順で $f(i) \leftarrow \max(f(i), f(i - 1))$ とすれば OK
- 最後に $i \leqq f(i)$ を反映しておく

4 f(l) を求める

- さっきの具体例なら
- $B_i = 2$ である i における C の列が (3, 7, 9) で
- $B_i = 3$ である i における C の列が (4, 6, 9)

- f(1) ← max(f(1), 3), f(4) ← max(f(4), 7), f(8) ← max(f(8), 9), f(10) ← max(f(10), inf)
- f(1) ← max(f(1), 4), f(5) ← max(f(5), 6), f(7) ← max(f(7), 9), f(10) ← max(f(10), inf)
- とあらかじめしておく

4 f(l) を求める

- 今の f の列は (4, -inf, -inf, 7, 6, -inf, 9, 9, inf, inf, ...) という形
- 累積 max を取ると (4, 4, 4, 7, 7, 7, 9, 9, inf, inf, ...) になる

- B ごとに C を集めて sort した列を作るのは全体で O(MlogM)

- あらかじめ max を取るのは O(M)
- 累積 max を取るのは O(P)

- f(l) を求める部分は O(MlogM + P) でできた

4 別の方法

- 累積 max を取らなくてもいける

- l が小さいものから求めるとして各列の中で l 以上の最小のものがどれかを管理しておく これらのうち最大の値が f(l)
- ある列の中で l 以上の最小のものが l ならその列の見るべき index を 1 増やす 同時に最大の値も更新しておく
- そのあと l ← l + 1 とする

- index を増やす回数は O(M) 回なので前のと同様 O(MlogM + P)
- (B, C) が同じ組がある場合の実装に注意

4 クエリの処理

- ここまで来ればクエリでは $f(L_j) \leqq R_j$ かを判定するだけ
- 全体で $O(MlogM + P + Q)$ で解けました

4 さらに別解

- 条件を観察すると、条件を満たさないときは B が同じ C を集めた列のうちどれかで $V_i < l \leqq r < V_{i+1}$ を満たしている
- つまり $[V_i + 1, V_{i+1} - 1]$ という形をした区間がたくさんあって各クエリでは区間 $[l, r]$ が与えられるのでたくさんある区間のどれかに包含されてないか判定してくださいというものになる
- どれかに包含されてるとき $V_i + 1 \leqq l \leqq r \leqq V_{i+1} - 1$ が成立
- つまり区間の左端が l 以下なら、区間の右端は r 未満なら OK

4 さらに別解

- 「区間の左端が l 以下なら、区間の右端は r 未満」を判定したい

- あらかじめ区間を左端で sort しておく これは O(MlogM)
- すると判定したいものは sort した後に x 番目までに出てくる区間の右端の最大値が r 未満かというもの

- この x は lower_bound などで求められる

- 累積 max をしておけば x が分かれば簡単

- 全体で O((M + Q)logM) で解けました

Subtask 5

$P \leqq 300\,000$

5 f(l) を求める

- f(l) を求めるのは小課題 4 の方法でできる
- これまでと同じように交換するフリーパスの l を固定すると O(PQ) になって間に合わない
- 交換する方法について考えてみる
- まず l を増やしたり r を減らすのは得しないので l を減らして r を増やす交換だけすればいい

5 交換について

- つまり (l, r) → (l - dl, r + dr) に変えると思う
- ここで 0 ≦ dl, dr とする

- f(l - dl) ≦ r + dr としたい
- dl + dr を小さくしたい

- 場合分けをする

5 交換について

- ① f(l - dl) ≦ r のとき

- dr = 0 として OK
- さらに、dl はそのような中で最小のものを取れば良い
- f は単調増加なので一度 f(l - dl) ≦ r になったら dl を増やしても同じ

- ② f(l - dl) > r のとき

- dr = f(l - dl) - r とすれば OK

5 クエリの処理 ①

- ① f(l - dl) ≦ r のとき

- このような最小の dl を見つけるにはどうすればいいか
- 言い換えると f(i) ≦ r なる最大の i を見つける問題になる

- これは lower_bound など配列上の二分探索を使えば OK

- よって ① にするパターンは O(logP) で解けた

5 クエリの処理 ②

- ② f(l - dl) > r のとき

- ① の考察を用いてこのような dl の範囲は分かる
- つまりある区間の dl における dl + dr = dl + f(l - dl) - r の最小値を知りたい

- 式変形をしていい形にしたい

- dl = - (-dl) と思うと f(l - dl) - (-dl) - r と変形できる
- -dl = (l - dl) - l と思うと f(l - dl) - (l - dl) + l - r と変形できる

5 クエリの処理 ②

- f(l - dl) - (l - dl) + l - r の最小値を求める問題になった
- g(x) = f(x) - x と思うと g(x) + l - r の最小値を求める問題に
- l, r は入力における L_j, R_j に対応しているのである区間における g(x) の最小値を求めればいい
- g を求めるのは簡単
- 区間における最小値を求めるといえば → **Segment Tree** や **Sparse Table** が使える

5 クエリの処理 ②

- Segment Tree なら構築 O(P) でクエリあたり O(logP)
- Sparse Table なら構築 O(PlogP) でクエリあたり O(1)
- クエリ部分は O(P + QlogP) ないし O(PlogP + Q) で可能
- 全体では O(MlogM + P + QlogP) や O(MlogM + PlogP + Q) で解けました
- f(i) + i や f(i) + A_i などを管理するテクニックはたまにみます

Subtask 6

$N, M \leqq 8\,000$

6 後ろの小課題も見てみよう

- この小課題は小課題 4 で使った条件の言い換えなしで解けます

- 後ろの小課題が前の小課題と独立していて、そちらが解けたということがある
- 後ろの小課題や問題も見よう！

6 f(l) を求める

- これまで同様に f を求めたいが l を P 通り試すと流石に大変
- そもそも P 通り試す必要はあるのか
- 例えば 2 本の辺について C が 4, 7 とあるとき 5 でも 6 でも 7 でも使える辺は変わらない
- つまり、l はある辺の C として登場するものだけ考えれば OK
- これは O(M) 通り

6 解法

- 小課題 3 で書いた尺取り法を使えば O((N + M)M) で f が分かる

- 次はクエリに答えるところ
- 小課題 5 では g(x) = f(x) - x として並べた

- 同様に g を求めて並べる

- 並べた中で何番目まで見ればいいかなどは同様にできる
- 座標圧縮をしている感じ

6 実装

- 実装するときにはどの値が何番目かを管理しておくと便利
- 全体としては O((N + M)M + QlogM) などで解ける
- 各 l について r の最小値を二分探索して O((N + M)MlogM) や O((N + M)MlogP) などがつくと間に合わないと思われる
- クエリあたり全部見て O(MQ) がついても間に合わないと思われる

Subtask 7

追加制約なし

7 小課題 5 と小課題 6 の復習

- 小課題 5 で f を求めるための言い換えと f を求めてからクエリに答えるところをやった
- 小課題 6 で座圧できることがわかった

- これを合体すれば OK

- 全体で O((M + Q)logM) で解けました

7 別解

- 交換をすることである i について [i, f(i)] を含むようにしたい

- [L, R] から伸ばすことを考える

- ① r を増やすだけ = 右端を伸ばす
- f(l) を求めた l のうち L 以上の中で最小のものに着目すれば OK

- ② l を減らすだけ = 左端を伸ばす
- f(l) ≦ r なる l のうち最大のものが分かれば OK

7 別解

- ③ l を減らして r も増やす ＝ 左端も右端も伸ばす

- このとき伸ばした後の区間は含みたい区間のどれかに一致
- そうでないなら余計に伸ばしたことになっている

- ではどの区間に一致させるか

- [i, f(i)] に一致させるとき i ≦ L ≦ R ≦ f(i) なので伸ばしたのは (伸ばしたあとの長さ) - (最初の長さ) ＝ (f(i) - i) - (R - L)

7 別解

- 結局ある範囲における f(i) - i を求める形になる
- 計算量は変わらず O((M + Q)logM) になりました
- 機械的に式変形しても別解のように区間を伸ばす形にしても結論は同じようになる

7 間違いについて

- 考察をすると区間を伸ばして与えられた区間のうちどれかを含むようにしたいという問題になった

- 与えられた区間を左端で sort して $T_1, T_2, ..., T_n$ とでも呼ぶ
- $T_i = [l_i, r_i]$ とでもして伸ばす区間を $S = [L, R]$ としよう

- $f(i) = S$ から T_i に持っていくためのコストとして求めたいのは $\min_i(f(i))$
- ここで予想「$f(i)$ は三分探索によって極値が計算できる」を立てる 実際に書いてみると通らない

7 間違いについて

- 実はこの予想は間違い 証明をする

- M = 2N - 3 として辺を C が小さい順に sort したとき B が (2, 3, ..., N, 2, 3, ..., N - 1) になるとする
- このとき $T_i = [C_i, C_{i+N-2}]$ になる

- N = 6 として具体例を構築する

7 間違いについて

i	1	2	3	4	5	6	7	8	9
B	2	3	4	5	6	2	3	4	5
C	3	5	X - 7	X - 5	X	X + 3	X + 4	2X - 1	2X

- 大きな数 X を用いて以下のようにしたとする
- S = [X, X] とする

- T は [3, X], [5, X + 3], [X - 7, X + 4], [X - 5, 2X - 1], [X, 2X]
- 考察より T の長さが大事 これは (X - 3, X - 2, 11, X + 4, X)

7 間違いについて

- (X - 3, X - 2, 11, X + 4, X) という列から最小値を見つけるのは今回の三分探索では正当かもしれないが少なくともこの列は凸ではない

- なので三分探索は正当ではない

Step 8

得点分布

8 得点分布(高 3 含む)

点数	100	80	71	58	51	49	48	42	38	30	26	23	19	15	8	7
1	o	o	o	o	-	o	o	-	o	o	o	-	-	o	-	o
2	o	o	o	o	o	o	o	o	o	-	o	-	o	o	o	-
3	o	o	o	o	o	o	o	o	-	-	o	-	o	-	-	-
4	o	o	o	o	o	o	-	o	o	o	-	o	-	-	-	-
5	o	o	-	o	o	-	-	-	-	-	-	-	-	-	-	-
6	o	o	o	-	-	-	o	-	-	-	-	-	-	-	-	-
7	o	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
人数	33	2	1	1	1	10	4	1	2	4	14	1	1	18	1	26
累計	33	35	36	37	38	48	52	53	55	59	73	74	75	93	94	120