JOI 2022/2023 春季トレーニング Day 3
旅行 (Tourism) 解説

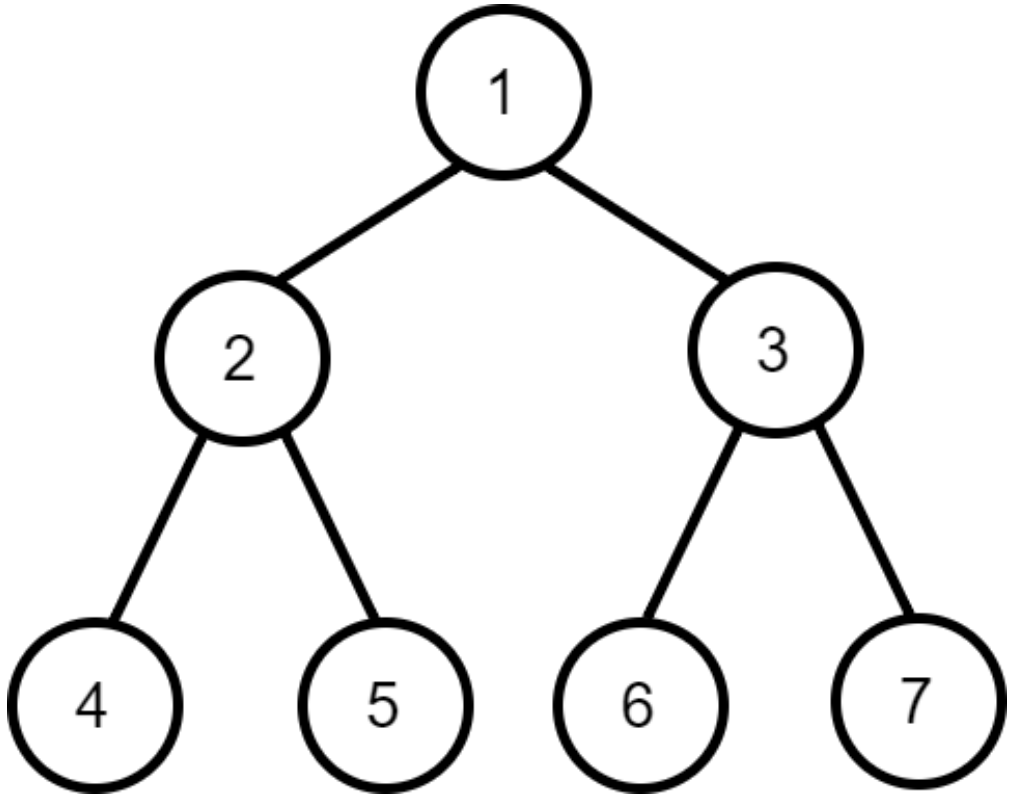
解説担当 : 菅井

問題概要

- N 頂点の木
- 木の頂点の列 $C_1, C_2, \ldots, C_M$
- Q クエリ: 各クエリでは L, R が与えられる
- 木上を移動し，頂点 $C_L, C_{L+1}, \ldots, C_R$ をすべて通りたい
- 1 回以上通った頂点の個数の最小値は？
- $N, M, Q \leqq 100\,000$

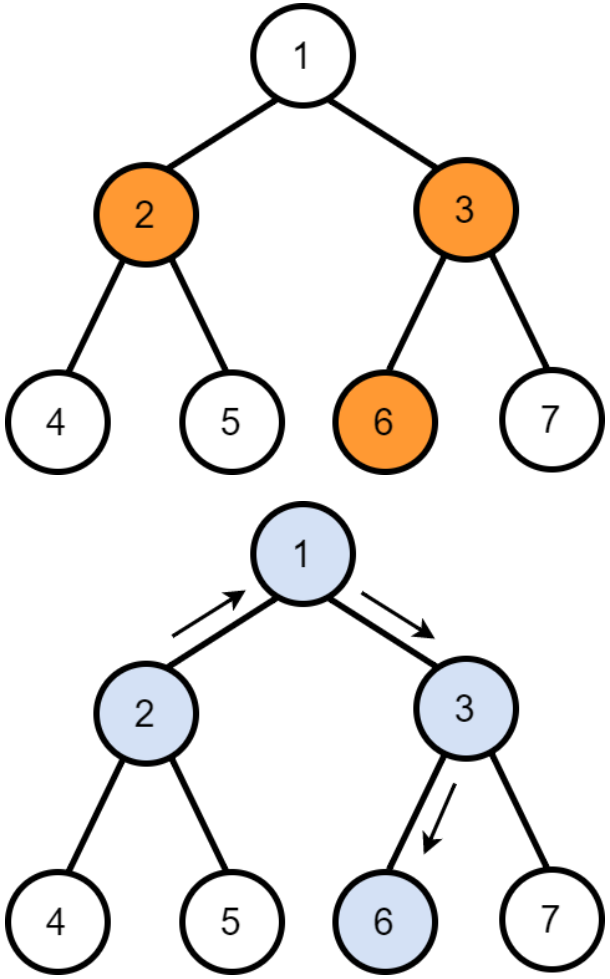
サンプル 1

- $C = [2, 3, 6, 4, 5, 7]$
- $L_1 = 1, R_1 = 3$
- $L_2 = 4, R_2 = 6$



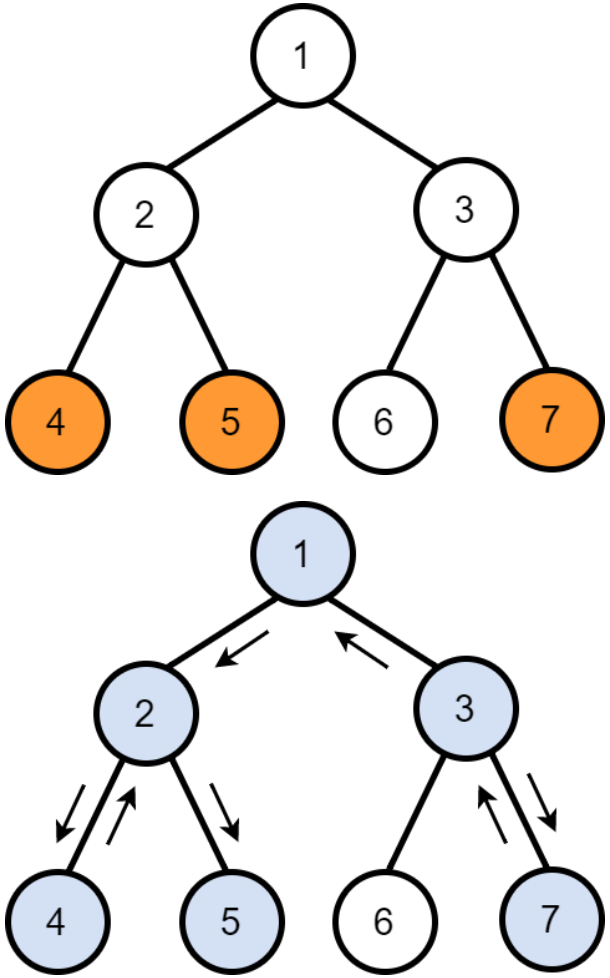
クエリ 1

- 頂点 $2, 3, 6$ を通りたい
- $2 \to 1 \to 3 \to 6$ と移動
- 4 つの頂点を 1 回以上通る



クエリ 2

- 頂点 $4, 5, 7$ を通りたい
- $3 \to 7 \to 3 \to 1 \to 2 \to 4 \to 2 \to 5$ と移動
- 6 つの頂点を 1 回以上通る



小課題

1. (5 点) $N, M, Q \leqq 300$
2. (5 点) $N, M, Q \leqq 2\,000$
3. (7 点) パス
4. (18 点) 区間が共通部分を持たない
5. (24 点) 完全二分木 (の一部)
6. (41 点) 追加の制約はない

小課題 1 (5 点)

頂点 $C_L, C_{L+1}, \ldots, C_R$ を通りたい

最終的に通る頂点の集合はどのようになるか？

考察

- $i = L, L+1, \ldots, R-1$ について，頂点 C_i と C_{i+1} を結ぶパス上の頂点は必ず通る必要がある
    - パス上のある頂点を通らないと，C_i と C_{i+1} を行き来できないので，必要な頂点をすべて通れない

(C_L - C_{L+1} パス) $\cup$ (C_{L+1} - C_{L+2} パス) $\cup \cdots \cup$ (C_{R-1} - C_R パス) に含まれる頂点は必ず通らなければならない

考察

逆に，これらの頂点だけ通る方法が存在する

- C_L から開始
- C_L から C_{L+1} へ最短経路で移動
- C_{L+1} から C_{L+2} へ最短経路で移動

  ⋮

- C_{R-1} から C_R へ最短経路で移動

小課題 1 (5 点) 解法

- (C_L - C_{L+1} パス) $\cup$ (C_{L+1} - C_{L+2} パス) $\cup \cdots \cup$ (C_{R-1} - C_R パス) に含まれる頂点の個数を求めればよい
- BFS/DFS など → 経路を復元する
- コーナーケース: $L = R$ の場合に注意 (答えは 1)
- 計算量 $O(NMQ)$

小課題 2 (5 点)

- 最終的に通る頂点は木をなす
- 頂点を数えるかわりに，辺を数えることにしてもよい (最後に 1 を足しておく)
- ある辺を通らなければならない条件は？

考察

- ある辺を切ると 2 つの木に分かれる
- 2 つの木どちらにも通りたい頂点があるとき，またそのときのみ，その辺を通る必要がある

典型: 根付き木にする

- 問題が根付き木でなくても，根を勝手に定めてしまうと実装が楽になることが多い
- BFS を使うと，各頂点の親・子・深さなどを簡単に計算できる
- BFS 順も持っておくと便利

```
std::queue<int> Q;
Q.push(0);
while (!Q.empty()){
  int v = Q.front();
  Q.pop();
  bfs.push_back(v);
  for (int w : E[v]){
    if (w != parent[v]){
      parent[w] = v;
      child[v].push_back(w);
      depth[w] = depth[v] + 1;
      Q.push(w);
    }
  }
}
```

考察

- 2 つの木どちらにも通りたい頂点があるとき，またそのときのみ，その辺を通る必要がある
- 辺が頂点 u, v を結ぶとし，u が v の親だとすると，v の部分木に通りたい頂点がないか，通りたい頂点がすべてあるなら，その辺は通らない
- そうでないなら，その辺は通る

小課題 2 (5 点) 解法

- 各頂点について，部分木に通りたい頂点が何個あるか木 DP で求める
- $\mathrm{dp}[v] = 0$ または $\mathrm{dp}[v] = R - L + 1$ なら，その辺は通らない
- そうでないなら，その辺は通る
- 計算量 $O((N + M)Q)$

小課題 3 (7 点)

- パスグラフ
- 通る頂点は区間になる
- $C_L, C_{L+1}, \ldots, C_R$ のうち最小の頂点から最大の頂点まで通る

小課題 3 (7 点) 解法

- 区間 $[L, R]$ の最小値と最大値をそれぞれ求めればよい
- Segment Tree を 2 本作る
- 計算量 $O(N + M + Q \log M)$

小課題 4 (18 点)

- クエリの区間に共通部分がない
- クエリに含まれる頂点数に比例する計算量で解くことができればよい

考察

- 結局，頂点 $C_L, C_{L+1}, \ldots, C_R$ を含む木の頂点数/辺数の最小値を求めればよい
- これは「最小シュタイナー木問題」で，一般のグラフでは NP 困難であることが知られている
- 木上の最小シュタイナー木といえば…?

典型: 木上の最小シュタイナー木

頂点 $X_1, X_2, \ldots, X_k$ を含む木の辺数の最小値は，以下のように求められる

- 前計算で DFS し，木の頂点を行きがけ順に並べる
- $X_1, X_2, \ldots, X_k$ を行きがけ順で現れる順にソートする
- $\frac{1}{2}\{\mathrm{d}(X_1, X_2) + \mathrm{d}(X_2, X_3) + \cdots + \mathrm{d}(X_{k-1}, X_k) + \mathrm{d}(X_k, X_1)\}$ が答え．ただし，$\mathrm{d}(u, v)$ は頂点 u, v の距離
- 類題: yukicoder #901 K-ary εxtrεεmε

小課題 4 (18 点) 解法

- 前計算で頂点を行きがけ順に並べる: $O(N)$
- 各クエリで頂点をソート: $O((R-L)\log(R-L))$
- 距離を $O(R-L)$ 回求める
- 合計で $O(N+M(\log N+\log M))$ など

小課題 5 (24 点)

- 完全二分木

考察

- (C_L - C_{L+1} パス) $\cup$ (C_{L+1} - C_{L+2} パス) $\cup \cdots \cup$ (C_{R-1} - C_R パス) に含まれる頂点の個数を求めたい
- $L = l$ と固定すると，R を増やしたときに集合から辺が取り除かれることはないので，それぞれの辺が含まれるかどうかは R について単調性がある
- 境界の位置を求めることを考える

考察

- C_l - C_{l+1} パスに含まれる辺は，$R \geqq l+1$ のときに含まれる
- それ以外の辺で，C_{l+1} - C_{l+2} パスに含まれる辺は，$R \geqq l+2$ のときに含まれる
- それ以外の辺で，C_{l+2} - C_{l+3} パスに含まれる辺は，$R \geqq l+3$ のときに含まれる

  ⋮
- 最後まで選ばれなった辺は R によらず含まれない

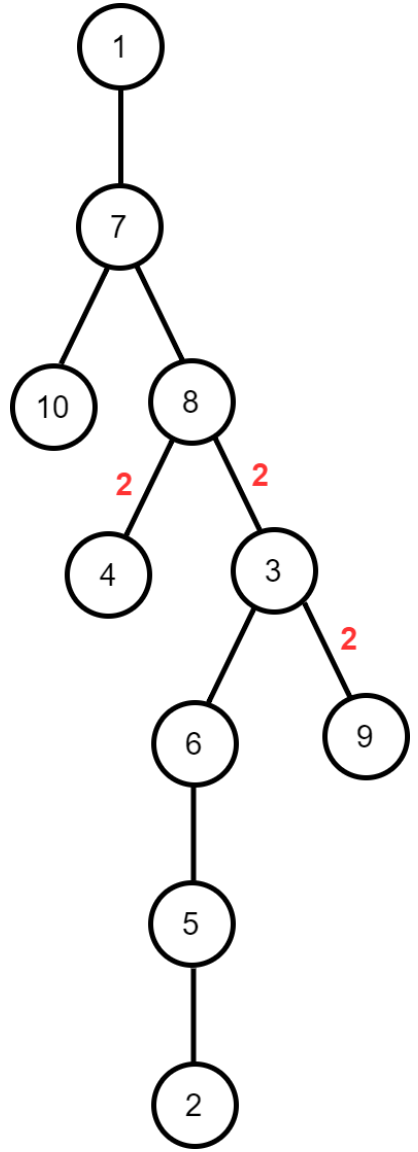
考察

それぞれの辺に，「その辺が含まれるような R の最小値」を書きたい．(絶対に含まれないなら ∞)

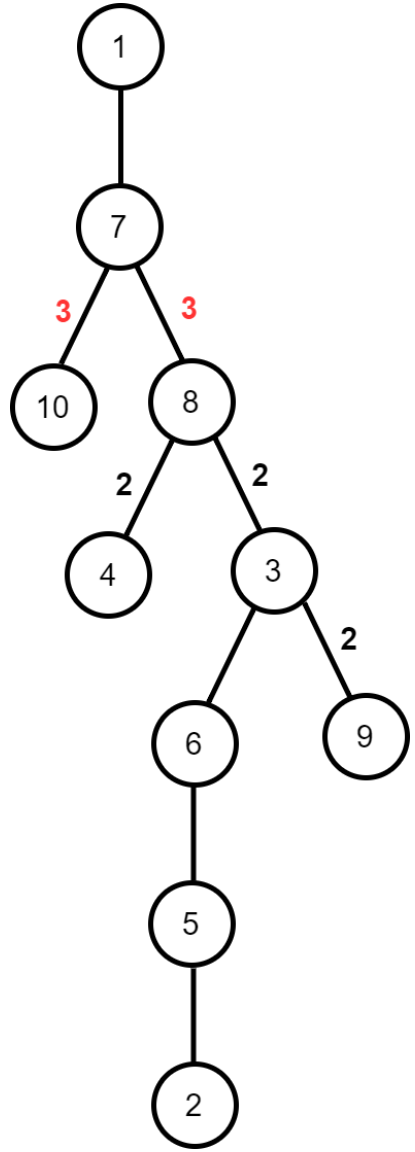
これは以下のような手順でできる

- $i = l+1, l+2, \ldots, R$ の順に，C_{i-1} - C_i パスに含まれるまだ何も書いていない辺に i と書く
- 最後まで何も書いていない辺は ∞ と書く

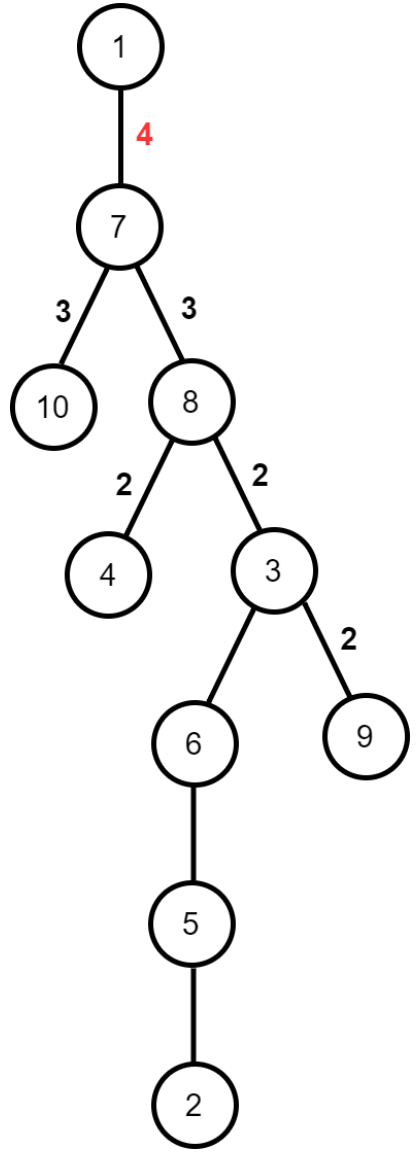
- $C = [9, 4, 10, 1, 10, 7, 6]$
- $l = 1$
- $(C_1 =)\ 9$ - $(C_2 =)\ 4$ パスに含まれる辺に 2 と書く



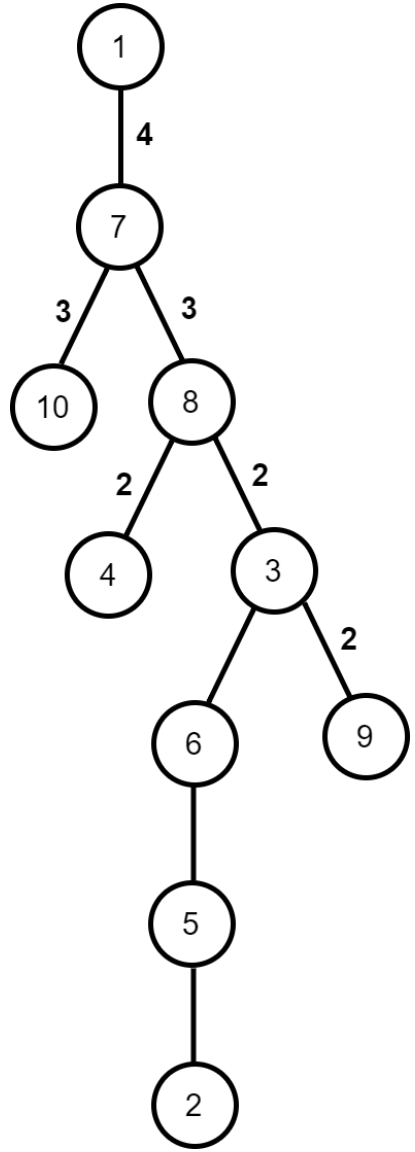
- $C = [9, 4, 10, 1, 10, 7, 6]$
- $l = 1$
- $(C_2 =)$ 4 - $(C_3 =)$ 10 パスに含まれるまだ何も書いていない辺に 3 と書く



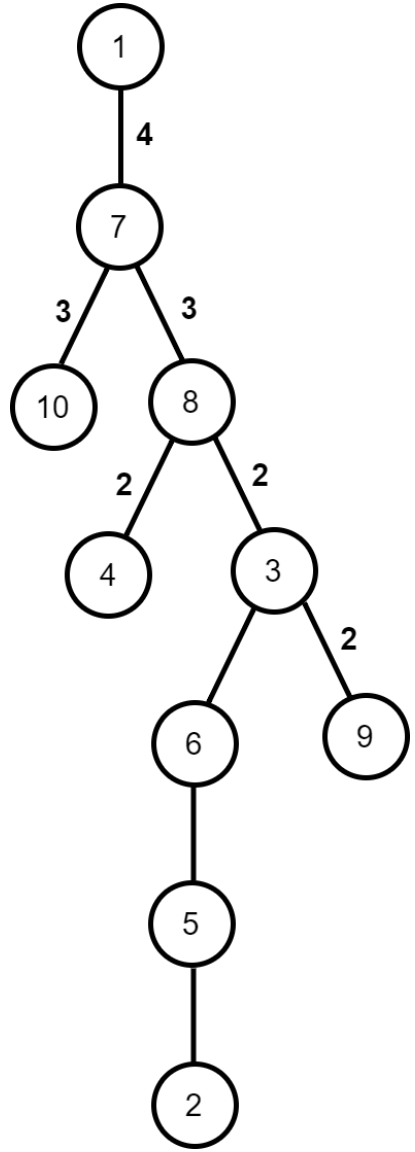
- $C = [9, 4, 10, 1, 10, 7, 6]$
- $l = 1$
- $(C_3 =)$ 10 - $(C_4 =)$ 1 パスに含まれるまだ何も書いていない辺に 4 と書く



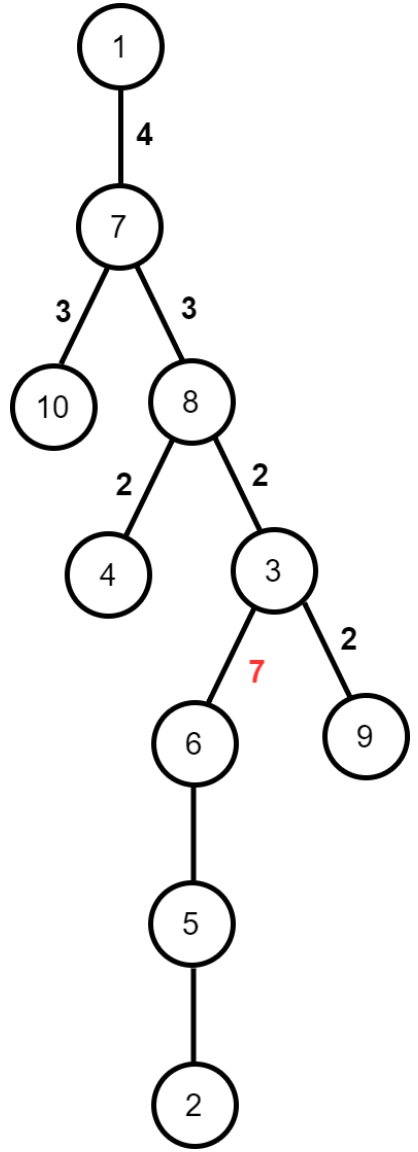
- $C = [9, 4, 10, 1, 10, 7, 6]$
- $l = 1$
- $(C_4 =)\ 1$ - $(C_5 =)\ 10$ パスに含まれるまだ何も書いていない辺に 5 と書く



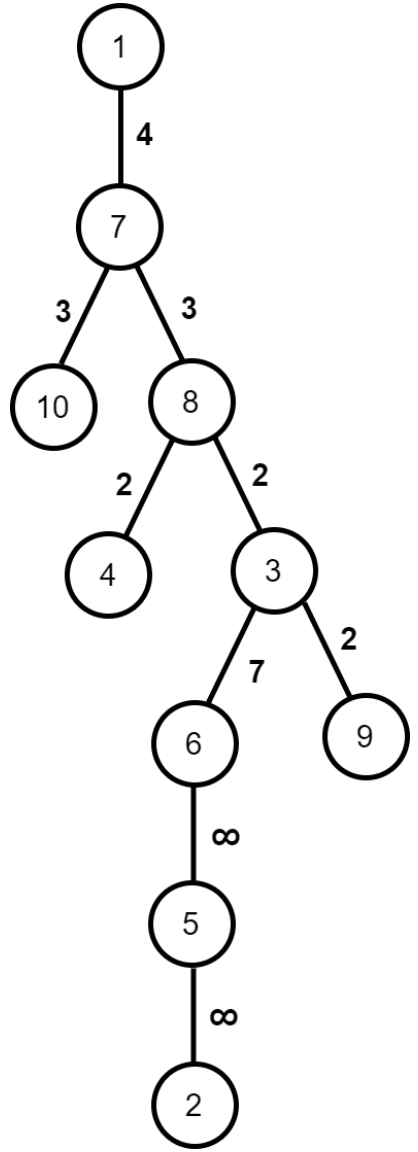
- $C = [9, 4, 10, 1, 10, 7, 6]$
- $l = 1$
- $(C_5 =)$ 10 - $(C_6 =)$ 7 パスに含まれるまだ何も書いていない辺に 6 と書く



- $C = [9, 4, 10, 1, 10, 7, 6]$
- $l = 1$
- $(C_6 =)$ 7 - $(C_7 =)$ 6 パスに含まれるまだ何も書いていない辺に 7 と書く



- $C = [9, 4, 10, 1, 10, 7, 6]$
- $l = 1$
- まだ何も書いていない辺に ∞ と書く
- ($R = r$ のときの答え) = (r 以下の値が書かれた辺の本数) $+1$



考察

- L が固定の場合は解けそう
- 実際は L が各クエリで異なるため，このままでは解けない
- 「まだ何も書いていない辺」は l によって異なるため，難しそう

典型: 更新クエリ

- 「区間を〇〇に更新する」というクエリは，逆から見ると「区間に〇〇と書き，それ以降変更されないようにする」クエリになる
- 「区間に〇〇と書き，それ以降変更されないようにする」は「区間のまだ何も書いていないところに〇〇と書く」と同じ
- 今回は逆に，更新クエリに置き換えて考える

言い換え前

- $i = l+1, l+2, \ldots, R$ の順に，C_{i-1} - C_i パスに含まれるまだ何も書いていない辺に i と書く
- 最後まで何も書いていない辺は ∞ と書く

言い換え後

- 最初にすべての辺に ∞ と書く
- $i = R, R-1, \ldots, l+1$ の順に，C_{i-1} - C_i パスに含まれるすべての辺に i と書く

この言い換えをしたうえで，l が変化したときどのように状況が変化するか考える

$l = k$ のとき

- すべての辺に ∞ と書く
- C_{R-1} - C_R パスに R と書く
- C_{R-2} - C_{R-1} パスに $R - 1$ と書く

$\vdots$

- C_k - C_{k+1} パスに $k + 1$ と書く

$l = k - 1$ のとき

- すべての辺に ∞ と書く
- C_{R-1} - C_R パスに R と書く
- C_{R-2} - C_{R-1} パスに $R - 1$ と書く

$\vdots$

- C_k - C_{k+1} パスに $k + 1$ と書く
- C_{k-1} - C_k パスに k と書く

- $l = k$ の状態から $l = k - 1$ の状態を得るには，C_{k-1} - C_k パスに k と書くだけでよい
- クエリを先読みし，L の降順に見る
- 最初それぞれの辺には ∞ と書かれている
- 以下の 2 種類のクエリを処理することに帰着
  - クエリ 1: 頂点 u, v と整数 x が与えられるので，u - v パスのそれぞれの辺に x と書く
  - クエリ 2: 整数 k が与えられるので，k 以下の数が書かれた辺の本数を答える

考察

- 小課題 5 では完全二分木なので，パスに含まれる辺の本数は $O(\log N)$
- 前半: パスに含まれる辺を列挙する
- 後半: それぞれの辺ごとに更新処理

前半

- パスに含まれる辺が少ないことがわかっているとき，それらを具体的に列挙したい
- 完全二分木なので，2 進数を用いてビット演算で計算することもできる
- 今回はもっと汎用的な方法を紹介する

典型: パスに含まれる辺の列挙

- 根付き木と考える
- 前計算で，各頂点の親と深さを求めておく
- 与えられる頂点を u, v とすると，$u = v$ になるまで以下を繰り返す
  - u と v のうち深さが大きいほうについて，自分とその親の間の辺を答え，親に置き換える (深さが同じ場合はどちらでもよい)
- $O(\text{パスの長さ})$ で列挙できる

後半

- $A[x] =$ (x が書かれた辺の本数) を管理する
- 1 本の辺に数を書くと，A の要素高々 2 つが変化する
- k 以下の数が書かれた辺の本数は，$A[1] + A[2] + \cdots + A[k]$ に等しい
- 結局，A 上の一点加算・区間和に帰着する
- Binary Indexed Tree を使えばよい

小課題 5 (24 点) 解法

- クエリを先読みし，L の大きい順に答える
- L を 1 減らすとき，C_{L-1} - C_L パスに含まれる辺を列挙し，BIT を更新する
- L がクエリの箇所まで来たら，$A[1] + A[2] + \cdots + A[R]$ を答える
- 計算量は $O(N + M \log N \log M + Q \log M)$

小課題 6 (41 点)

- 小課題 5 の考察から，以下の 2 種類のクエリが処理できればよい
  - クエリ 1: 頂点 u, v と整数 x が与えられるので，u - v パスのそれぞれの辺に x と書く
  - クエリ 2: 整数 k が与えられるので，k 以下の数が書かれた辺の本数を答える

考察

まずパスグラフの場合を考える

- 辺に書かれた数を並べ，数列を作る
- クエリ 1: l, r, x が与えられるので，区間 $[l, r)$ を x に更新する
- クエリ 2: 整数 k が与えられるので，k 以下の要素の個数を答える

考察

- 隣接する 2 つの要素が異なる箇所に仕切りを入れる
- 数列を連長圧縮して扱うと，区間に入っている仕切りの個数が少なければ高速に処理できる
- いわゆる「区間を `set` で持つテク」

実装

- $\left($区間の開始位置,値$\right)$の `pair` を `set` に入れる
- 両端に番兵を置くと実装しやすい
- 区間が消えるときは，その値の出現数が区間の長さだけ減る
- 最後に x の出現数が $r - l$ だけ増える
- 小課題 5 と同様に，それぞれの数の出現回数を BIT で管理

計算量解析

- クエリ 1 で新たに最大 2 箇所に仕切りが入る
- 区間が消える回数は，区間が作られる回数で上から抑えられる → 区間が消える回数は $O(M)$ 回
- 全体で計算量は $O(N + (M + Q) \log M)$

考察

- パスの場合は解けたので, どうにかして問題をパスの場合に帰着させたい
- 木をパスに変換するアルゴリズム
- Heavy Light Decomposition

満点解法

- クエリを先読みし，L が大きい順に処理
- HLD して各 C_{i-1} - C_i パスを $O(\log N)$ 個の区間に変換し，区間ごとに処理
- 各数の出現回数を管理する BIT を更新
- 計算量 $O(N + M \log N \log M + Q \log M)$

別解 1: 分割統治

- 数列 C 上で分割統治
- 区間 $[l, r)$ を見ているとすると，$m = \left\lfloor \frac{l+r}{2} \right\rfloor$ とし m が含まれるクエリだけ考えればよい
- $((C_L \text{ - } C_{L+1}) \cup \cdots \cup (C_{m-1} \text{ - } C_m)) \cup ((C_m \text{ - } C_{m+1}) \cup \cdots \cup (C_{R-1} \text{ - } C_R))$ に含まれる辺の本数を求めたい
- $C_l, C_{l+1}, \ldots, C_r$ について木を圧縮し，$O(r - l)$ 頂点の木を作る

- それぞれの辺が $(C_L$ - $C_{L+1}) \cup \cdots \cup (C_{m-1}$ - $C_m)$ に含まれるかどうかは L について単調性がある (R についても同様)
- 各辺 e について，「$L \leqq x$ または $y \leqq R$」の条件と重みがあり，満たしている条件の重みの総和を求めることに帰着
- 平面走査と BIT でできる
- 計算量 $O(N + M \log^2 M + Q \log M)$ など

別解 2: Mo's Algorithm

- 小課題 4 で，DFS 順の位置を管理して，1 つ追加・削除を処理する
- 直前・直後の頂点の取得，距離の計算で $\log$ が付く
- 計算量 $O(N + M\sqrt{Q}\log N + Q\log Q)$: TLE
- Rollback Mo + 連結リストで直前・直後の頂点を取得し，Sparse Table で距離を計算すると $\log$ が落ちて満点

得点分布

得点	小課題	人数	累積人数
100	123456	1	1
59	12345	4	5
41	123 5	1	6
35	1234	6	12
31	3 5	1	13
28	12 4	1	14
17	123	7	21
12	1 3	3	24
5	1	1	25
0		2	27