F : Tree

原案 : tubo28

解答 : tubo28

解説 : tubo28

謝罪

- UVa 1674 とほぼ同じでした
  - 重みの頂点への加算

概要

- 根付き木がある
- 次のクエリを高速に処理せよ
  - addクエリ : v以下の辺の重みを全てw大きくする
  - distクエリ : u, vの距離を求める
- 制約は大きい
  - 頂点数 N <= 300,000
  - クエリ数 Q <= 300,000
    - AOJでcin,coutとscanf,printfの差は0.5秒くらいでした

TLE解

- クエリごとにO(N),全部でO(NQ)
- max(NQ)≒10^11なのでとても無理

TLE解

```
// v以下の辺にwを足す
void add(v,w):
    for c in {vの子}:
        辺(v,c)にwを足す
        add(c,w)
```

TLE解

```
// u,vの距離を求める
integer dist(u,v):
    if(visited[u]) return ∞
    if u==v:
        return 0
    res = ∞
    for x in {uと隣接する点}:
        res = min(res, 辺(u,x)の重み+dist(x,v))
    return res
```

準備1 LCA

- Lowest Common Ancestor
- 最低共通先祖問題
- lca(u,v) = u,vの共通する先祖で最も低い位置にある頂点

準備1 LCA

- LCAが求まると距離が求めやすくなる
  - lca(u,v) = x とすると
    - dist(u,v) = dist(u,x) + dist(v,x)
    - uからx, vからxへ行くには上に登るだけ
- 詳しくは
  - http://abc014.contest.atcoder.jp/tasks/abc014_4
  - http://www.slideshare.net/chokudai/abc014
  - 蟻本2版 292ページ

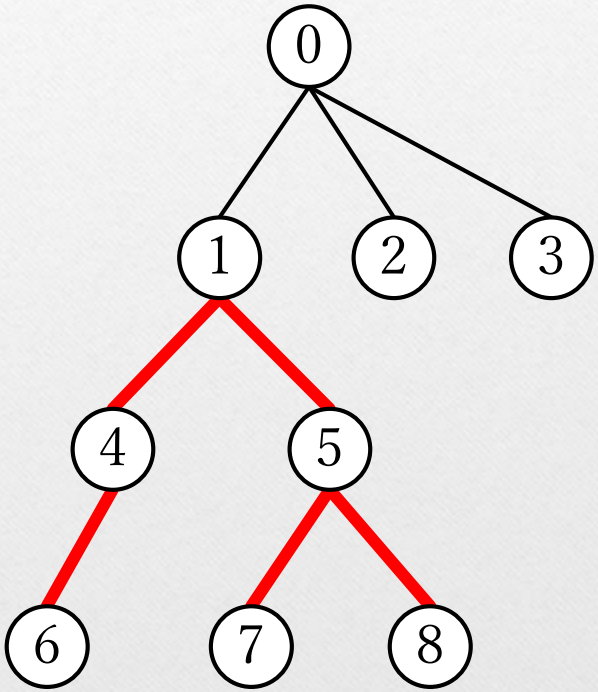
準備2 セグメント木

- 今回は距離が動的に変化する
- クエリは
  - 木上の区間に一様に足す
  - 木上の区間の和
- のどちらか

準備2 セグメント木

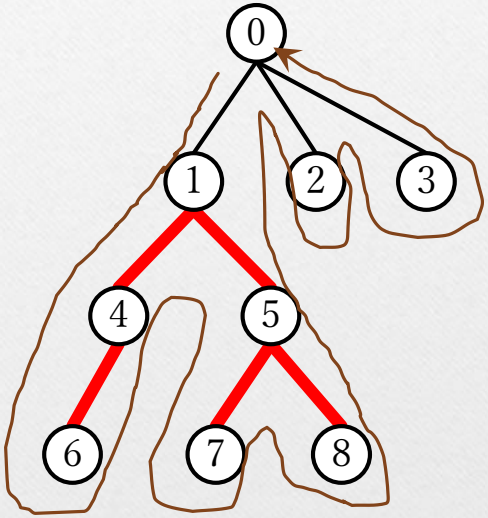
- 区間に対する処理が得意なデータ構造といえばセグメント木
  - http://www.slideshare.net/iwiwi/ss-3578491
  - 蟻本2版 153ページ
- 影響される辺をどうやって区間に並べるか
  - 区間？

addクエリ



- add(1,2)
- 赤が影響される辺

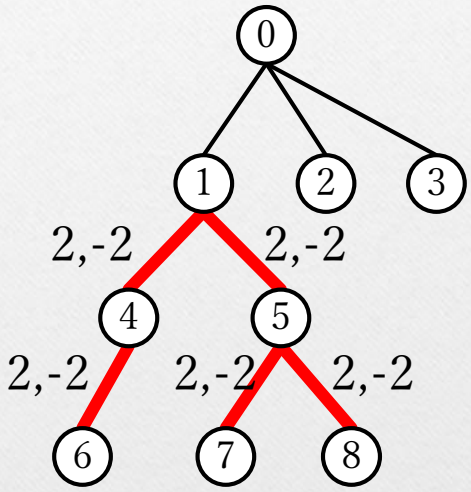
addクエリ



- 根からDFSする
  - EulerTour
- DFSで通った順に辺をリストに追加していくと…
- addの対象の辺が一列に並ぶ
  - その区間に一様にwを足せば良い
  - 1の最も左の子へ降りてから右の子から登ってくるまでが対象

(0,1)(1,4)(4,6)(6,4)(4,1)(1,5)(5,7)(7,5)(5,8)(8,5)(5,1)(1,0)(0,2)(2,0)(0,3)(3,0)

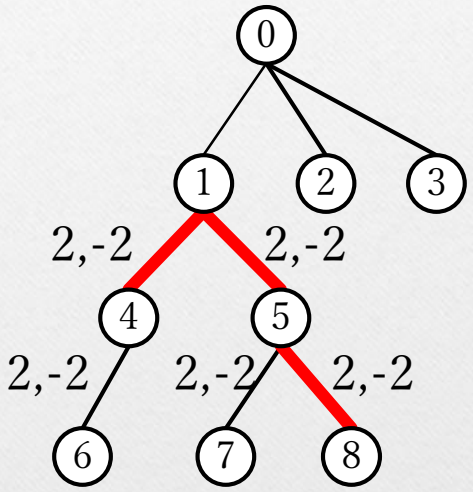
addクエリ



- ただし足すときに
  - 下りなら+
  - 登りなら－
- にしておく
- distクエリのため

(0,1)(1,4)(4,6)(6,4)(4,1)(1,5)(5,7)(7,5)(5,8)(8,5)(5,1)(1,0)(0,2)(2,0)(0,3)(3,0)

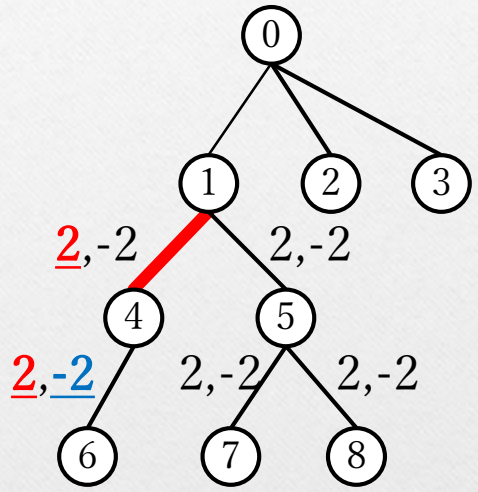
distクエリ



- dist(4,8)
  - lca(4,8) = 1 を求める
  - LCAで分けdist(1,4)+dist(1,8)と考える

(0,1)(1,4)(4,6)(6,4)(4,1)(1,5)(5,7)(7,5)(5,8)(8,5)(5,1)(1,0)(0,2)(2,0)(0,3)(3,0)

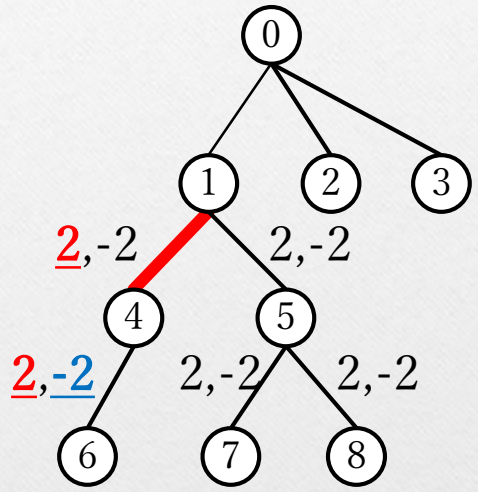
LCAからのdistクエリ



- dist(1,4)を求める
  - DFSで1を降りてから4に登ってくるまでに通る辺の重みの和を求める
  - 余計に足される辺は＋と－が両方含まれるのでうまく打ち消される

(0,1)(1,4)(4,6)(6,4)(4,1)(1,5)(5,7)(7,5)(5,8)(8,5)(5,1)(1,0)(0,2)(2,0)(0,3)(3,0)

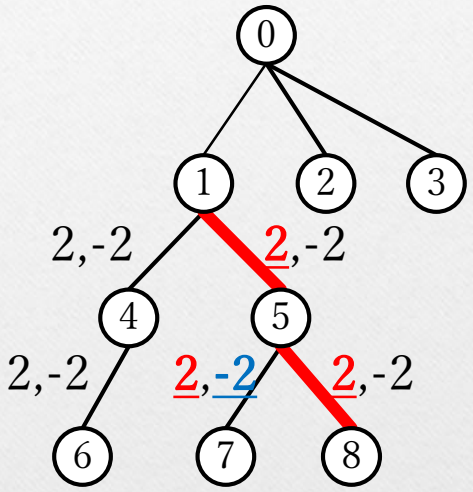
LCAからのdistクエリ



- 4より下の辺は＋－両方含まれる
- パス上の辺は＋だけが含まれる

(0,1) (1,4)(4,6)(6,4)(4,1)(1,5)(5,7)(7,5)(5,8)(8,5)(5,1)(1,0)(0,2)(2,0)(0,3)(3,0)

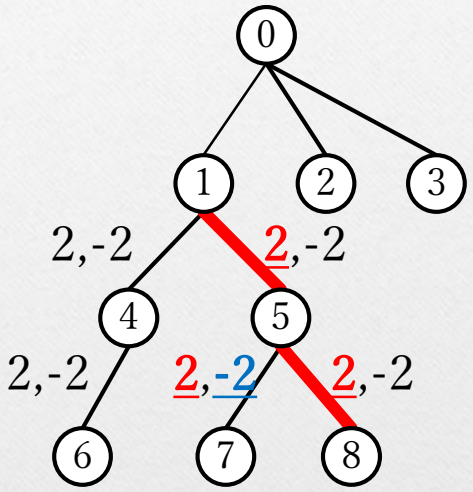
LCAからのdistクエリ



- dist(1,8)を求める
  - DFSで1を降りてから8に登ってくるまでに通る辺の重みの和を求める
  - 余計に足される辺は＋と－が両方含まれるのでうまく打ち消される

(0,1)(1,4)(4,6)(6,4)(4,1)(1,5)(5,7)(7,5)(5,8)(8,5)(5,1)(1,0)(0,2)(2,0)(0,3)(3,0)

LCAからのdistクエリ



- 8より下の辺は＋－両方含まれる(無いけど)
- パス上での5が含まれない枝への分かれ先の辺は＋－両方含まれる
- パス上の辺は＋だけが含まれる

(0,1)(1,4)(4,6)(6,4)(4,1)(1,5)(5,7)(7,5)(5,8)(8,5)(5,1)(1,0)(0,2)(2,0)(0,3)(3,0)

実装

- セグメント木を2本持つ
  - プラス用とマイナス用

計算量

- LCAにO(logN)
- addクエリにO(logN)
- distクエリにO(4logN)
  - 木を2本使った場合
- 全部でO(QlogN)
- TL4秒は厳しかったようで申し訳ないです
  - あまり考えていなかった
  - O(NQ)の嘘探索を落とせる程度に調整すべきでした

コーナーケース

- DFSでスタックオーバーフロー
  - ジャッジもハマった
  - ケースに含めるか迷った
    - Stackを使った方法への書き換えはそんなに難しくなかったので含めました
- 32bitでオーバーフロー
  - 和の最大値はNQmax(w)になる

想定外の解法

- Heavy-Light Decomposition (zerokugiさん)
  - ライターは知りません
- Separator Decomposition (antaさん)
  - ライターは知りません
- http://topcoder.g.hatena.ne.jp/iwiwi/20111205/1323099376

結果

- 全体
  - AC / submit
    - 6 / 119
  - FA
    - yutaka1999さん (88 min)
- オンサイト
  - AC / submit
    - 0 / 10
  - FA
    - さん (min)