Non-redundant Drive

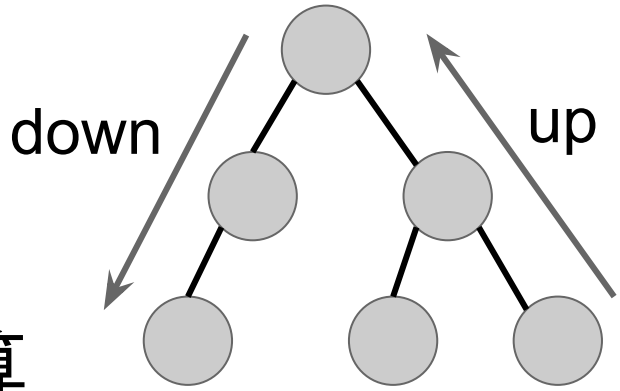
原案：宮村
問題文：井上(Darsein)
解答：井上(Darsein)・水野(not)
入力：井上(Darsein)
解説：井上(Darsein)・水野(not)

問題概要

- N頂点の木があり、辺の長さはd_i
- 各頂点ではg_iだけ燃料が補給できる
- 車は1単位の燃料で1単位の距離を進む
- 任意の頂点から燃料0の状態でドライブを開始
- 車は途中で燃料が負になると動かなくなる
- 同じ頂点や辺を利用してはならない
- 最大いくつの頂点を訪れることができるか

解法

- ある頂点vに注目してvを通るパスすべてについて考える
  - vに入るパス+vから出るパス



- 2つの配列を利用して最長を計算
  - up[i] : vの部分木で長さi以上のvで終わる列のうち、vの時点で残せる燃料の最大値
  - down[i] : 燃料が負でも車が動くと考えたとき、vの部分木で長さi以上のvから始まる列のうち、途中での燃料の最小値の最大値
  - vでの最長 = max(i+j | up[i]+down[i] ≥ 0)

解法

- Weighted Union Heuristics + 平衡二分探索木
  - データ構造のありがたいご利益でTLEするDPを加速する
- 重心分解
  - 重心を通るパスについて計算する

Weighted Union Heuristics

- いわゆる「データ構造をマージする一般的なテク」
- 2つのデータ構造を併合するとき、小さい方のデータ構造から大きい方へデータを移すようにマージする
- なぜか計算量が log に落ちる
  - 小さい方のデータ構造はマージ後のデータ構造の大きさの半分以下
  - 半分ずつ減るので、各要素は log 回しかデータ構造に現れない

Weighted Union Heuristics

- 部分木からup, downをいっぱいもらう
- 一番大きい部分木の up, down をベースにする
- 他の部分木を順々に最大のup, downにマージしていく
- マージする前に答えの計算をする
  - up, down は減少列になることがわかる
    → 二分探索できる
  - O(N log N) 回 O(log N) の二分探索をする
    → $O(N \log^2 N)$

Weighted Union Heuristics

- 他の部分木を順々に最大のup, downにマージしていく
- マージする前に答えの計算をする $O(N \log^2 N)$
- マージする際には以下が必要
  - 値を先頭に挿入 (up, 根で補給して移動なし)
  - 値の検索 (減少列なので二分探索できる)
  - 値の範囲 add (upの要素全てを $-d_e + g_v$ する)
  - 値の範囲更新
    (downの $-d_e$ より大きいprefixは$-d_e$にする)

Weighted Union Heuristics

- 他の部分木を順々に最大のup, downにマージしていく
- マージする前に答えの計算をする $O(N \log^2 N)$
- マージする際には以下が必要
  - 値を先頭に挿入 (u デックでO(1) 移動なし)
  - 値の検索 (減少列な 配列 O(logN) できる)
  - 値の範囲 add (up 配列 O(N) を $-d_e + g_v$ する)
  - 値の範囲更新 配列 O(N)
    (downの $-d_e$ より大きいprefixは$-d_e$にする)

Weighted Union Heuristics

- 他の部分木を順々に最大のup, downにマージしていく
- マージする前に答えの計算をする $O(N \log^2 N)$
- マージする際には以下が必要
  - 値を先頭に挿入 (減少し)
  - 値の検索 (減少
  - 値の範囲 add (
  - 値の範囲更新
    (downの $-d_e$ より大きいprefixは$-d_e$にする)

平衡二分探索木でO(logN)
平衡二分探索木でO(logN)
平衡二分探索木でO(logN)
平衡二分探索木でO(logN)

平衡二分探索木

- 二分探索木: 左の部分木に自分より小さい値、右の部分木に自分より大きい値を持つ二分木
  - 今回は列を管理するので、左の部分木には前の要素、右の二分木には後ろの要素
- 平衡二分探索木
  - 二分探索木へのクエリの計算量はだいたい O(深さ)
  - ただの二分探索木は偏って深くなると死 (最悪 O(N))
  - 深くなったらよしなに調整してバランスをとる
    → 平衡二分探索木
  - 有名な実装として、赤黒木, Treap, Splay木, など

遅延評価平衡二分探索木

- 普通の平衡二分探索木でできるクエリ
  - 検索, (1つの) 値の更新
  - 追加/削除, 併合/分割, etc.
- 今回欲しい機能
  - 範囲add, 範囲更新
    → 遅延評価で実装
- 範囲addの例: 足したい要素をいったん根で止めておいて、他のクエリで下に降りるときに足して、ついでに降ろす
- 詳しい実装は http://www.slideshare.net/iwiwi/2-12188757 などを参照

解法(1) の雑なまとめ

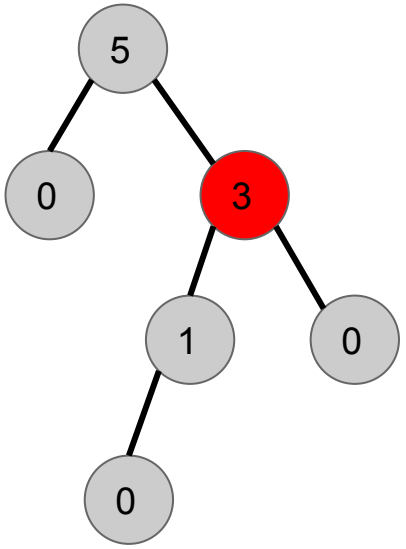
- up, down配列を部分木からもらって更新する木DPをする
  - 愚直にやると $O(N^2)$
- マージ・答えの計算の際に Weighted Union Heuristics を用いることで一部 $O(N \log^2 N)$ に
- マージで必要なクエリを速くするために配列をやめて平衡二分探索木にすると完全に $O(N\log^2 N)$に

解法2（重心分解）

- 頂点vを通るすべてのパスについて計算する
  - down[i] : それぞれの子についてDFSする
  - up[i] : それぞれの子についてDFSする
- upをなめて、downについて二分探索する
- downは今まで見た子についてマージする
- 子を見る順番を逆にして同じことをもう一度する
- 木のサイズをNとしてO(N log N)

重心分解

- 木の重心とは全ての部分木の大きさがN/2以下になる頂点
- DFSをして戻ってくるときに、各頂点の子孫の数を計算
- 初めてN/2以上になる頂点が重心
- 再帰的に重心で分割を繰り返すと分割の深さがO(logN)になる



解法2のまとめ

- 重心分解をする
- 重心を通るすべてのパスについてupをなめてdownを二分探索する
- $O(N \log^2 N)$

ジャッジ解

- 井上：290行 (マージテク + BST)
- 水野：119行 (重心分解)

結果

- First AC：yutaka1999 (180:01)
- Accepted：2 / 4