KUPC 2019 L – タケノコ

writer: kazuma

問題概要

- 各頂点にタケノコが生えた根付き木がある（最初は根だけ）
- タケノコは高さ x と伸びやすさ y を持ち、効果 a の肥料をまくと高さが +ay 伸びる
- **オンラインで3種類のクエリ処理**
  - 頂点追加
  - 全タケノコに肥料散布
  - 根から頂点 v へのパス上で、それまでより真に高いタケノコの数（ただし30より多い場合は数えなくてよい）

一次式とみなす

- まず、タケノコの高さを一次式と見なす

- 変数 w = 0 を用意して、全体に肥料 a が散布されると w += a
- すると、タケノコの高さは (w – (追加された時点でのw)) * y + x と表せる

- これは w の一次式

- Q. 一次式が出てくるってことは Convex Hull Trick とか使ったら解けるんでは？（適当）

- A. 実は正しい

Convex Hull Trick とは

- 一次式 f(x) = ax+b を追加したり x の値を渡すと f(x) の最大値を取得したりできるデータ構造です

- 追加される一次式の傾き a が単調増加/減少の場合のみで使えるものなどもありますが、ここでは一般の場合で使えるものを考えます

各頂点にCHTを持つ

- 各頂点に、根からその頂点までのパス上に含まれる全てのタケノコの一次式を持ったCHTを持つ
- さらに深さの情報も使うと、根から各頂点までのパス上で最も高いタケノコのうち深さが小さいものの位置がわかる
- そして、その位置に移動するのを繰り返すと、実はクエリ3で数えたいものを下から数えることができる

- Q. そんなにCHTを持ったらMLE&TLEするのでは？

- A. CHTを(省メモリ&高速に)完全永続化すればよい

完全永続CHT

- ということで完全永続なCHTがほしい！
  - でも平衡二分木で動的にやるやつだと、永続化したら計算量が壊れるので無理！
- これは実は Li Chao Tree という最近流行り(要出典)のデータ構造をベースにすると解決
  - 追加も取得も最悪 O(log(値の範囲))
  - ほとんどセグメントツリーなので永続化は簡単
- くわしくは「Li Chao Tree」とか「Li Chao Segment Tree」で検索すると出てくると思うので省略

まとめ

- クエリ1:頂点vのCHTに一つタケノコを追加したCHTを生成
  - O(logX) (Xはタケノコの高さの範囲)
- クエリ2:w += a
  - O(1)
- クエリ3:最大30回CHTに質問クエリ
  - O(30 logX)
- 全体 O(Q1 logX + Q2 + 30 Q3 logX)

別解

- 平方分割する
  - オンラインなのに平方分割できるんですか？
  - 確率的に平方サイズになるように分割する
  - 各パケットはdisjointに作るのではなく
    ある頂点の直近 平均O(sqrt(Q)) 程度の数の頂点を含むようにする
- 各パケットでCHTを構成する
- クエリは各パケットを見ていき、
  - 前から見えるタケノコがあれば愚直に計算
  - なければ次のパケットを見る
- だいたい O(Q sqrt(Q) (logQ+30)) 程度の時間計算量で解ける