ICPC Summer camp 2012 Day 4

問題 D
Do use segment tree

原案：森
解答例：森、平澤
問題文：平澤
解説：森

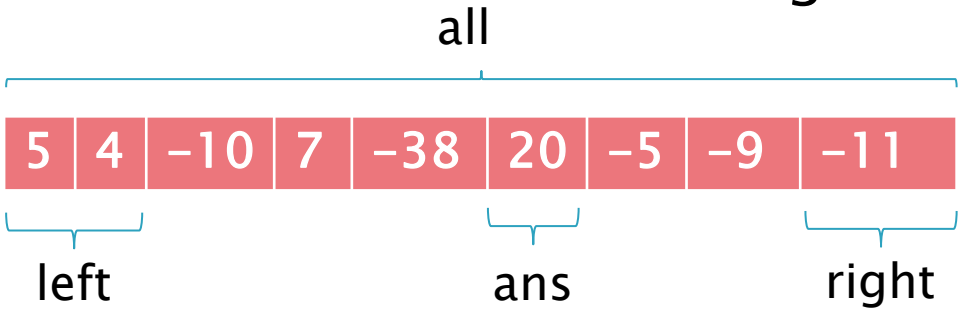
問題

- 頂点数nの木Gが与えられる
- 各頂点のコストはw_i
- 以下の2種類のクエリが合計q個あるので捌け
  - パスa–bに含まれる頂点の重みをcに変更する
  - パスa–bに含まれる頂点の重みを列にした時の部分和の最大値を出力せよ
- $1 \leq n \leq 200{,}000$
- $1 \leq q \leq 100{,}000$
- $|w_i|, |c| \leq 10{,}000$

解法 (数列の場合)

- とりあえず数列で考えてみる
  - さらに更新はa-aのみの場合
- クエリの見た目からどうせSegment Treeで解けそう

- Segment Treeの各ノードに必要そうな値を考える
  - その区間の最大値ansは絶対必要
  - その区間の合計allも必要そう
  - 左端を含んだ最大値left、右端を含んだ最大値rightがあれば便利



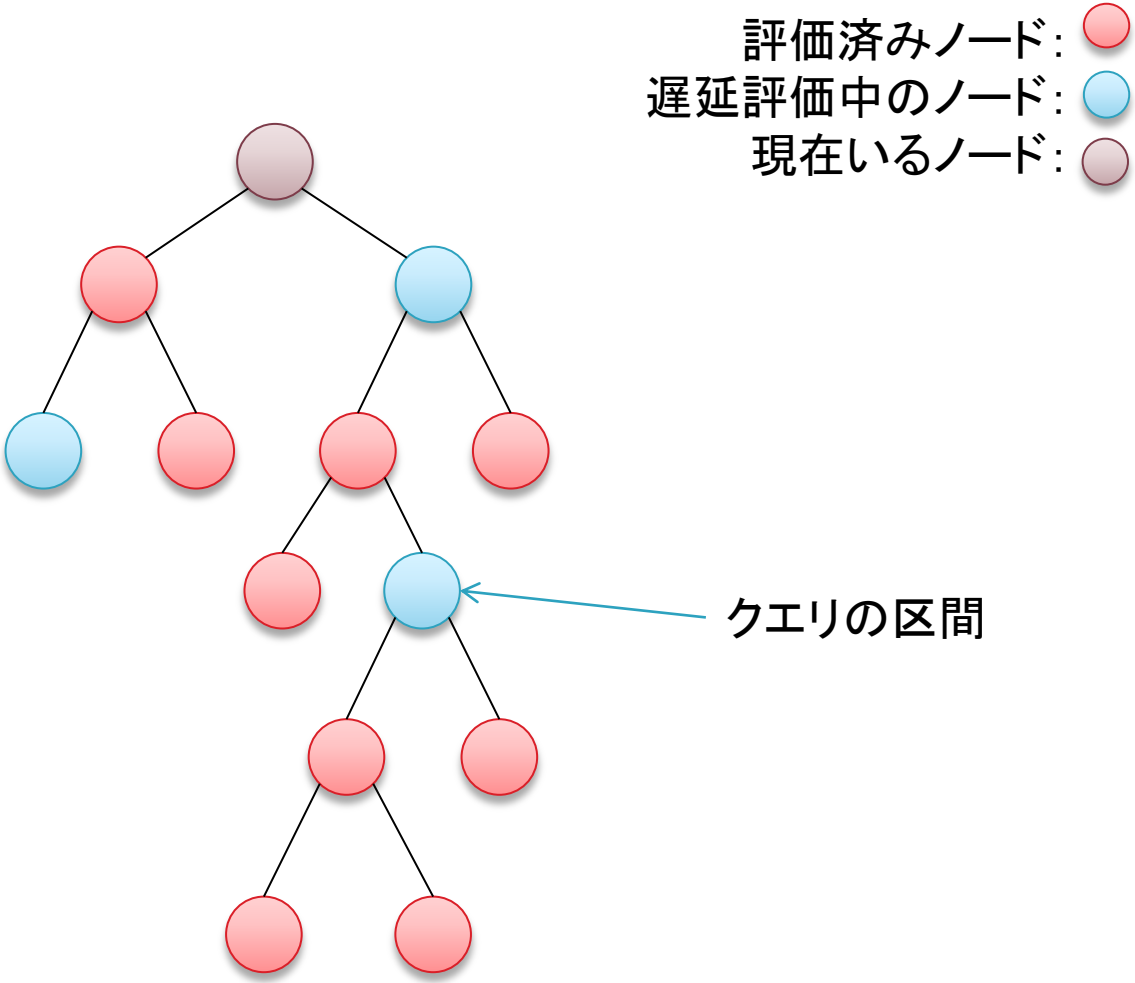
解法 (数列の場合)

- ans,all,left,rightがあれば十分
- Segment Treeの性質から隣り合った区間のノードからマージされた区間のノードが作れれば良い
  - m.ans = max(l.ans, r.ans, l.right + r.left)
  - m.all = l.all + r.all
  - m.left = max(l.left, l.all + r.left)
  - m.right = max(r.right, r.all +l.right)
    - ただし、lが左の区間、rが右の区間、mがマージされた区間

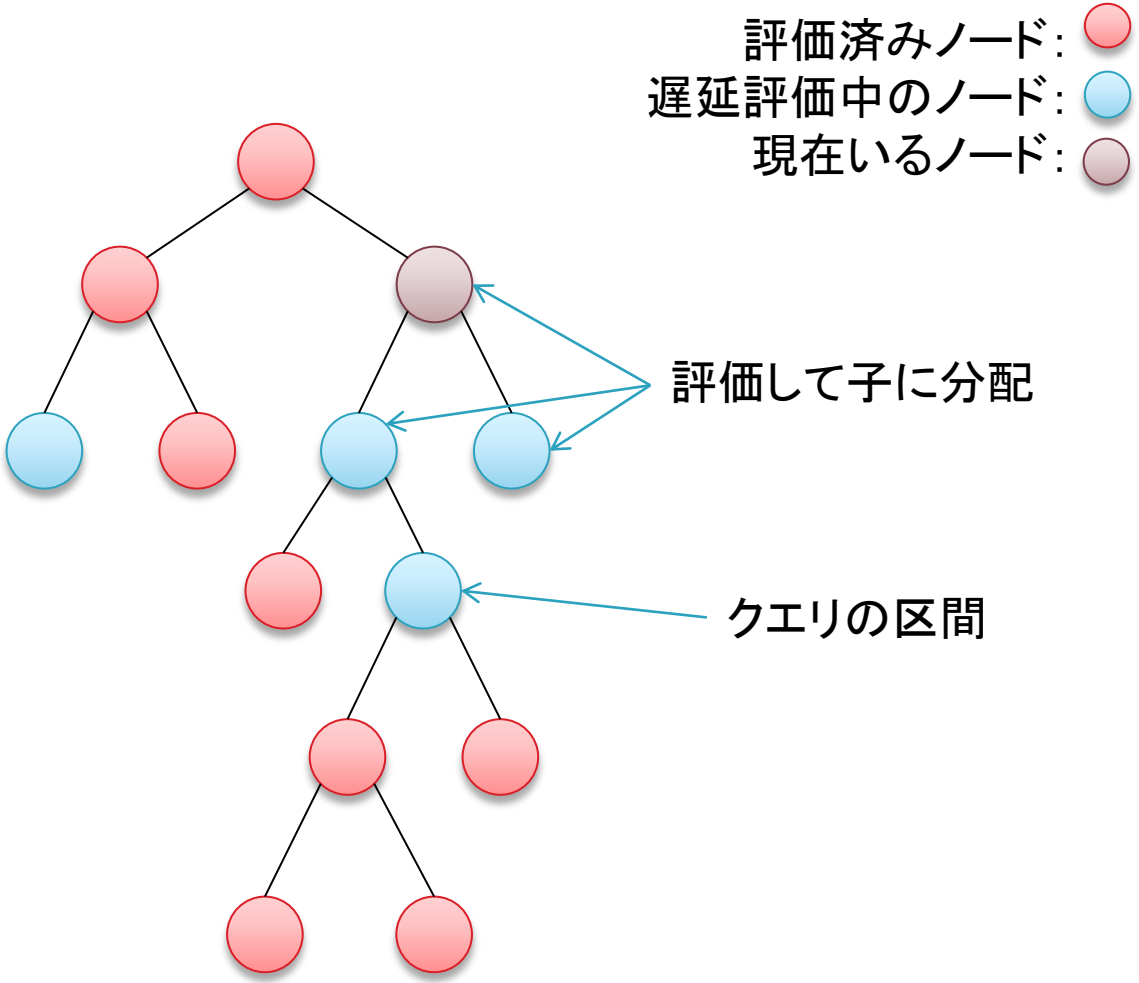
解法 (数列の場合 with 範囲更新)

- 範囲更新があると値の更新が$O(\log n)$でできない？
- 遅延評価を使おう！
  - 更新する区間とピッタリ合う場所とその祖先だけを更新
  - 更新した区間を通る時は1つ下の子だけを前もって更新

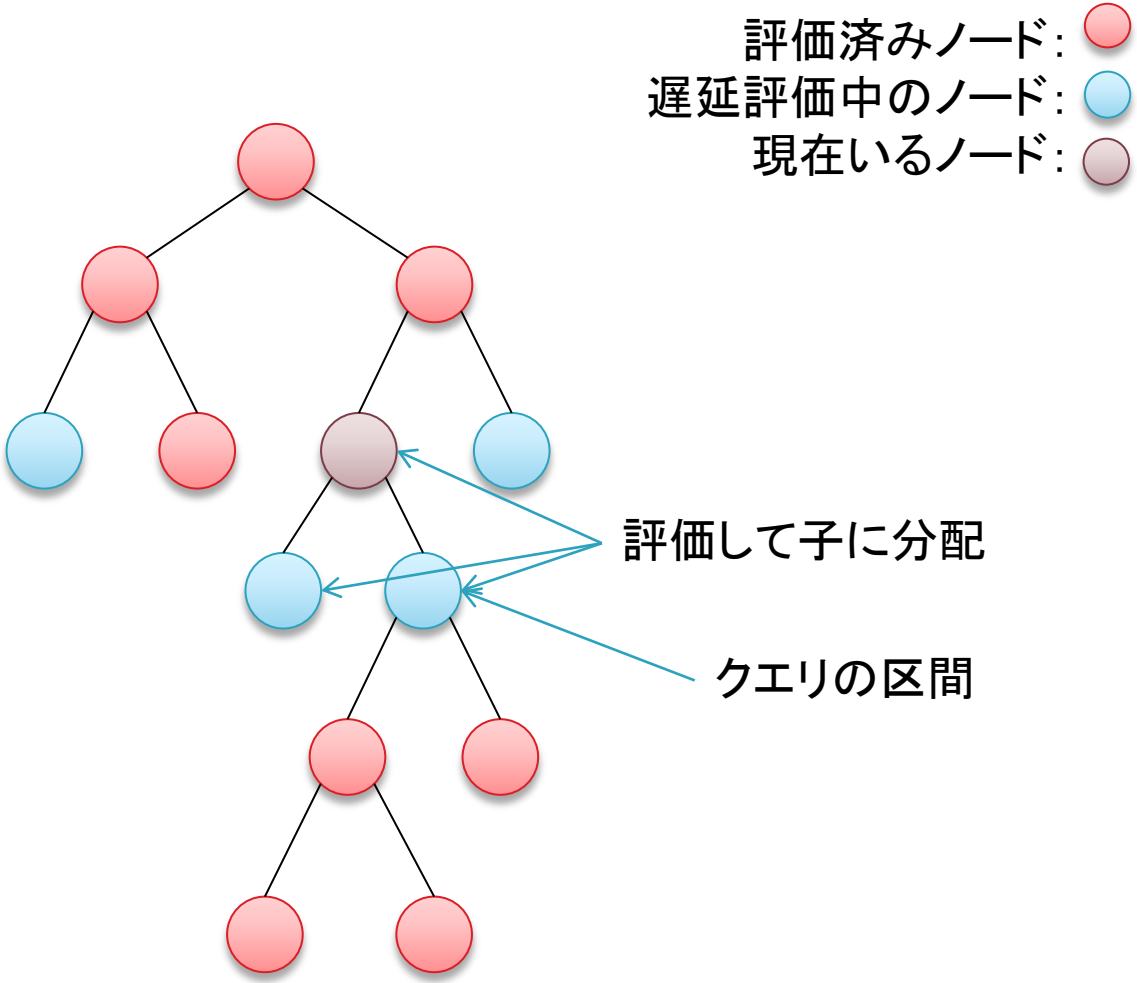
遅延評価の例



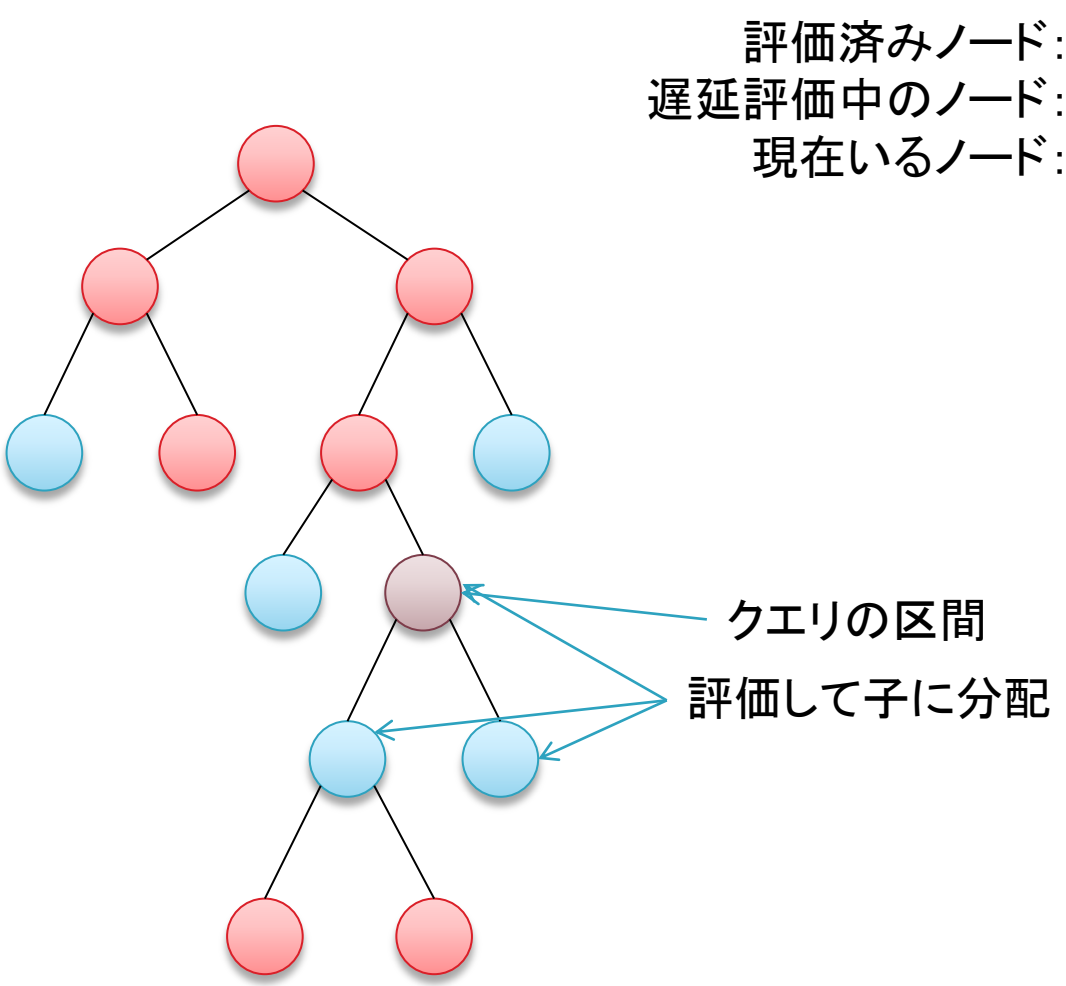
遅延評価の例



遅延評価の例

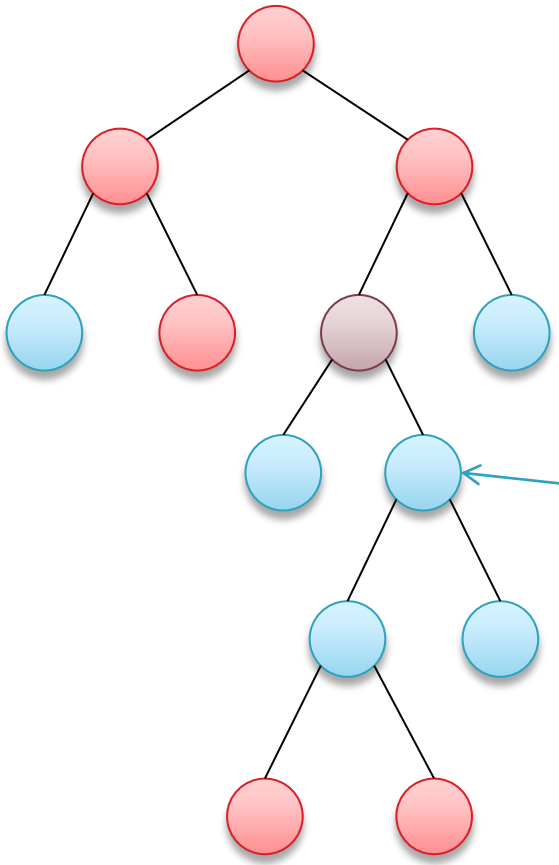


遅延評価の例



遅延評価の例

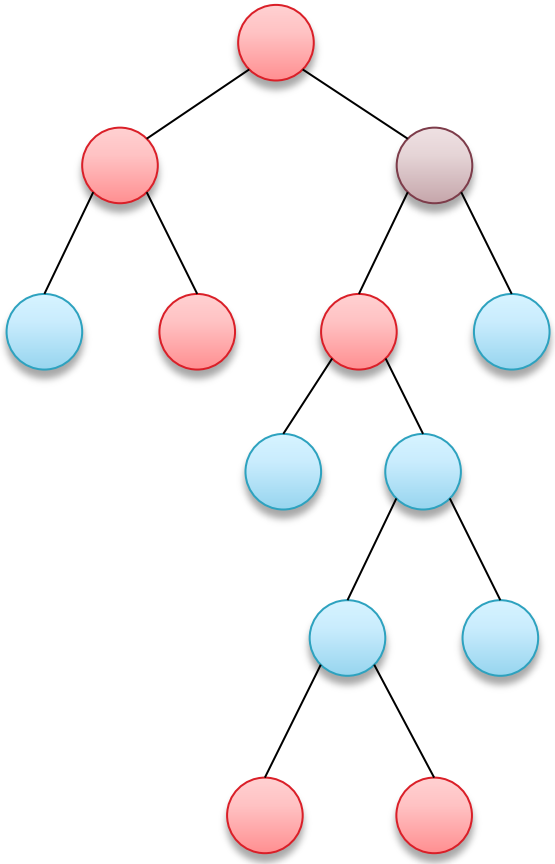
評価済みノード:
遅延評価中のノード:
現在いるノード:



遅延評価のフラグを立てて
マージしながら戻る

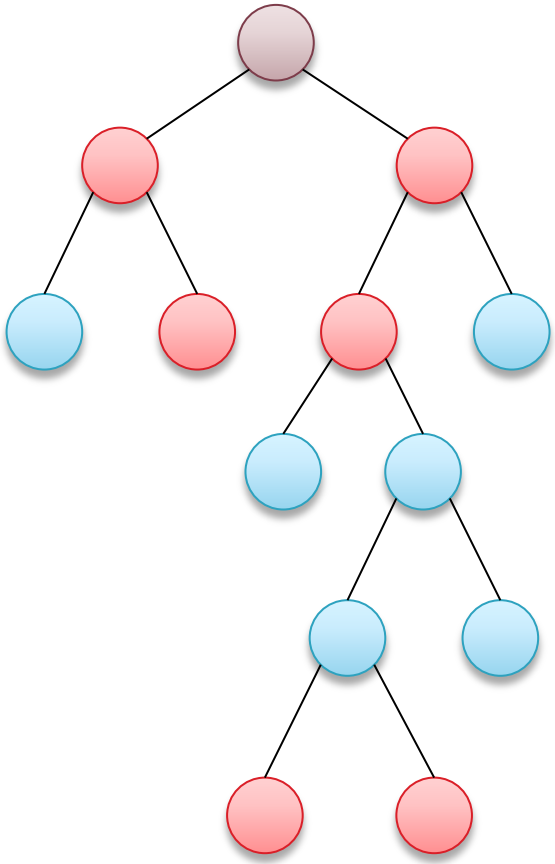
遅延評価の例

評価済みノード:
遅延評価中のノード:
現在いるノード:



遅延評価の例

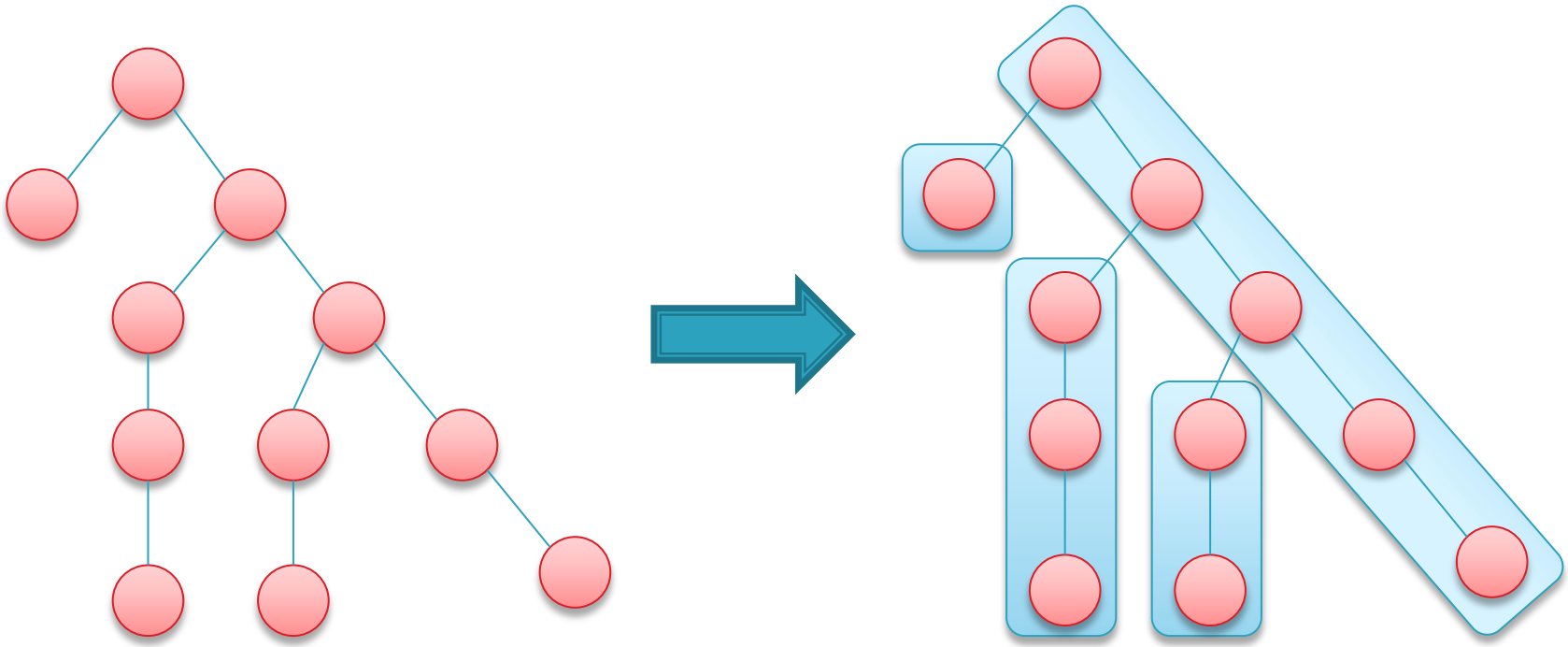
評価済みノード:
遅延評価中のノード:
現在いるノード:



木の場合

- 数列ができたので木の場合を考える
  - 木が列みたいなので考えられるとよさそう
  - Heavy Light Decompositionを使えば良い
- 木を複数の鎖に分割するアルゴリズム
  - 部分木のサイズが最もでかいやつをHeavy Edgeで結んで鎖にする
  - 他はLight Edgeで結んで再帰的に処理
  - 再帰の深さは$O(\log n)$
- 格鎖に対してSegment Treeを持たせれば、クエリを高速に処理可能
  - 同じ鎖になるまで区間をマージしながら深い方を上にのぼらせていく感じで

Heavy Light Decomposition



まとめ

- 木をHeavy Light Decompositionする
- 各鎖に対してSegment Treeを対応付ける
- 各クエリに対してパスを見つけSegment Treeから答えを計算
  - Segment Treeは遅延評価で範囲更新する
- 計算量は$O(n + q \log^2 n)$
  - がんばれば2乗は消せるかもね

- 実装はたいへん
- 動的木を使っても解けるかも

注意点

- Heavy Light Decompositionする際に部分木の深さで計算すると再帰の深さが$O(\sqrt{n})$になる入力が存在する
- グラフをdfsするとスタックオーバーフローする
  - 8MBだと再帰10万回はぎりぎりセーフかも、それ以上だと死
- 出力のクエリに答えるときに一番上の鎖での処理に注意
  - Segment Treeのノードのleft,rightを片方逆にしないとダメ

ジャッジ解

- 森
  - 352行 9600B
- 平澤
  - 323行　10500B

結果

- First AC
  - mithril (189分)
- AC / Submit
  - 1 / 6 (17%)
- AC / Trying people
  - 1 / 2 (50%)