JOI Open 2014 Day1 Factories

概要

- 辺にコストがある木が与えられる
- 次のクエリをたくさん処理せよ
  - disjoint な頂点集合 X，Y が与えられるので，X に含まれる点から Y に含まれる点まで移動する時の距離の最小値を求めよ

小課題 1

- N, Q <= 5000
- クエリごとに O(N) が許される

- 木 DP
- (その点から最も近い X, Y 中の点までの距離) を返しながら DP をすると答えが出せる

- あるいは，Dijkstra を実行しても O(QN log N) で，間に合う

小課題 2

- 各 S, T <= 20
- クエリごと O(ST) が許される

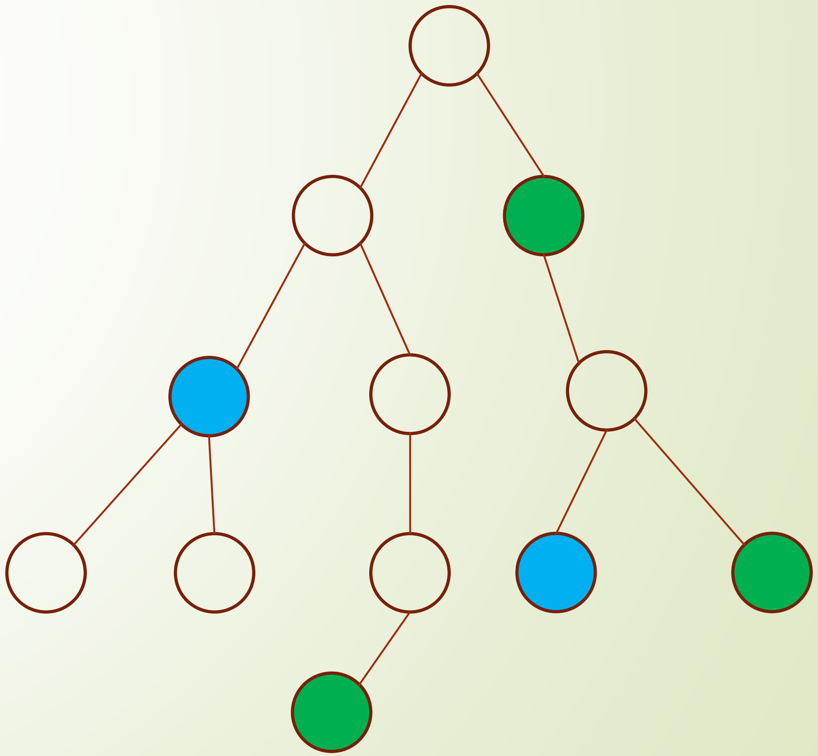
- すべての点対を試せば良い
- 根付き木にして，各点の根からの距離を求めておき，LCA を O(1) で求められる準備をしておくと，2 点間の距離が O(1) で求められる

満点解法

- S, T の和がそれぞれ 1,000,000 以下である
- 入力サイズの線形より速くは解けない
- O(S+T) とか O((S+T) log N) くらいで解ければよさそう

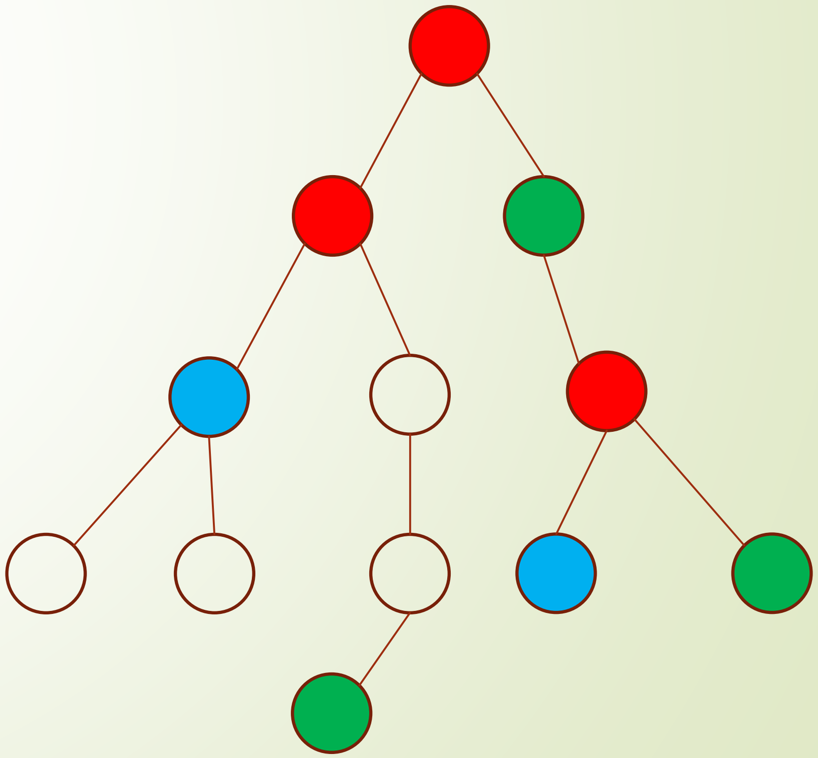
観察

- 木 DP をするときに，N 個の頂点すべて見ないといけない？
- X の点を青で，Y の点を緑で表す



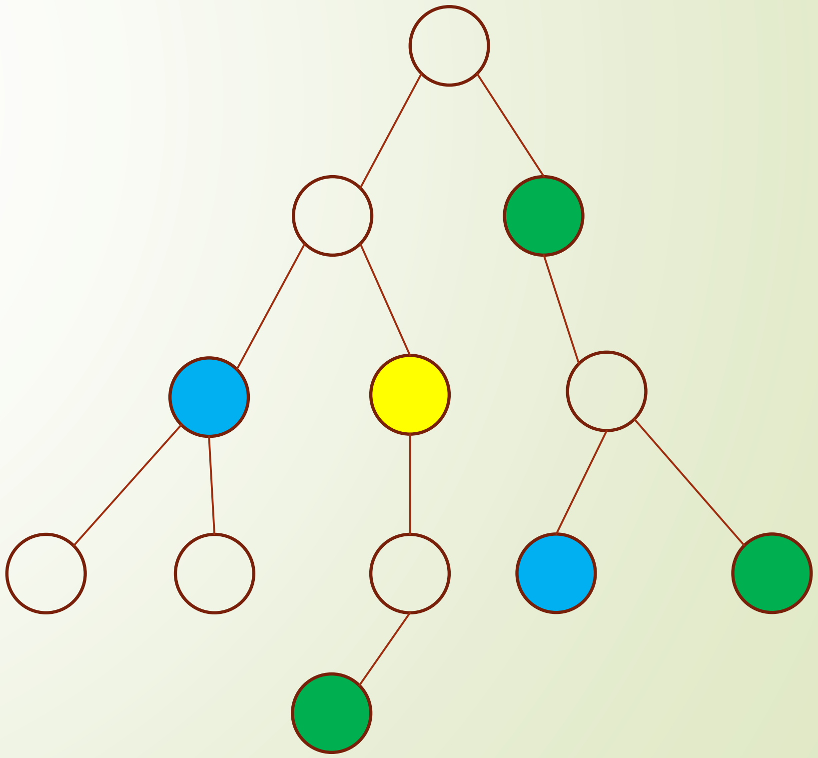
観察

- 木 DP をしたいだけだったら，必要な頂点は
  - X, Y に最初から含まれる点
  - 最初から含まれる点の LCA となる点
- だけ (右図では，赤い頂点)
- 他の点を調べても，
  - 下に X, Y に含まれる点がない
  - 下から来た値をそのまま受け流す
- のどちらかで，無意味



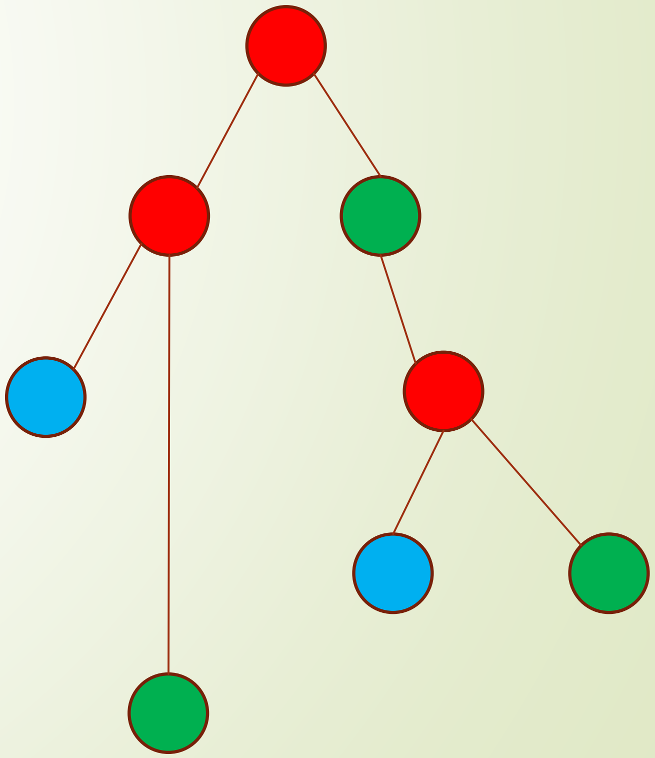
観察

- たとえば，黄色の点は，DP をしたところで，直下の点の答えに適当に辺の長さを足して返すだけなのでほとんど無意味



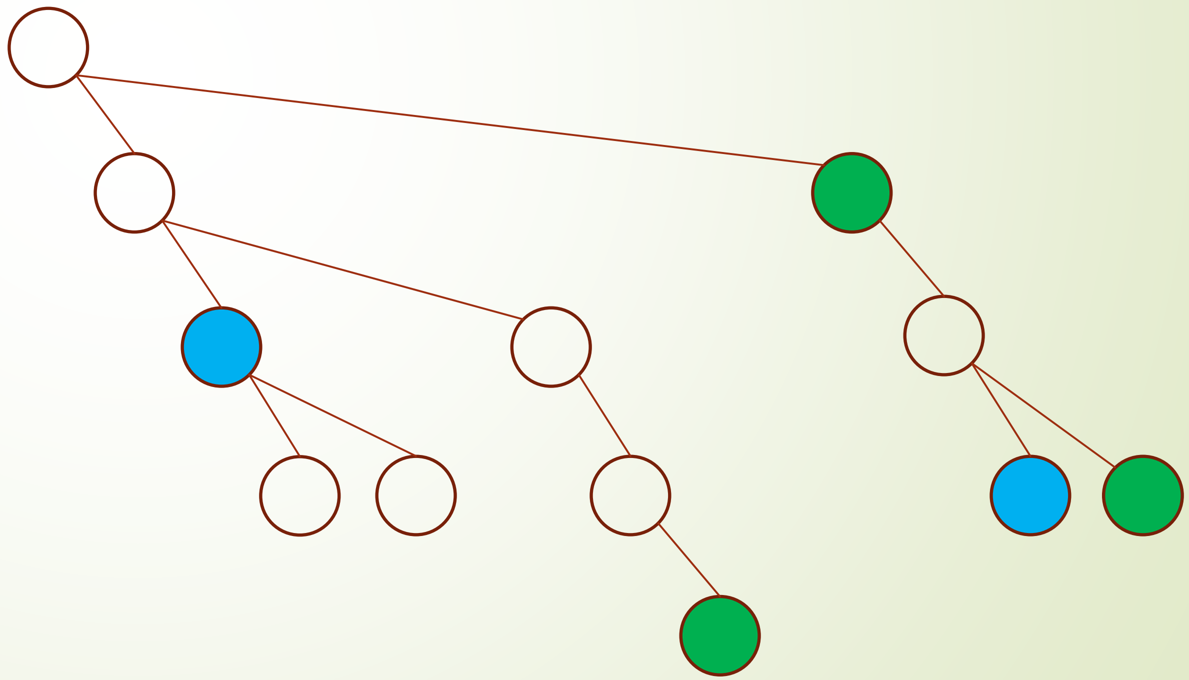
観察

- 結局，右のグラフについて DP すればよくなる
- 点をつぶしたとき，辺の長さは単に和を求めればよい



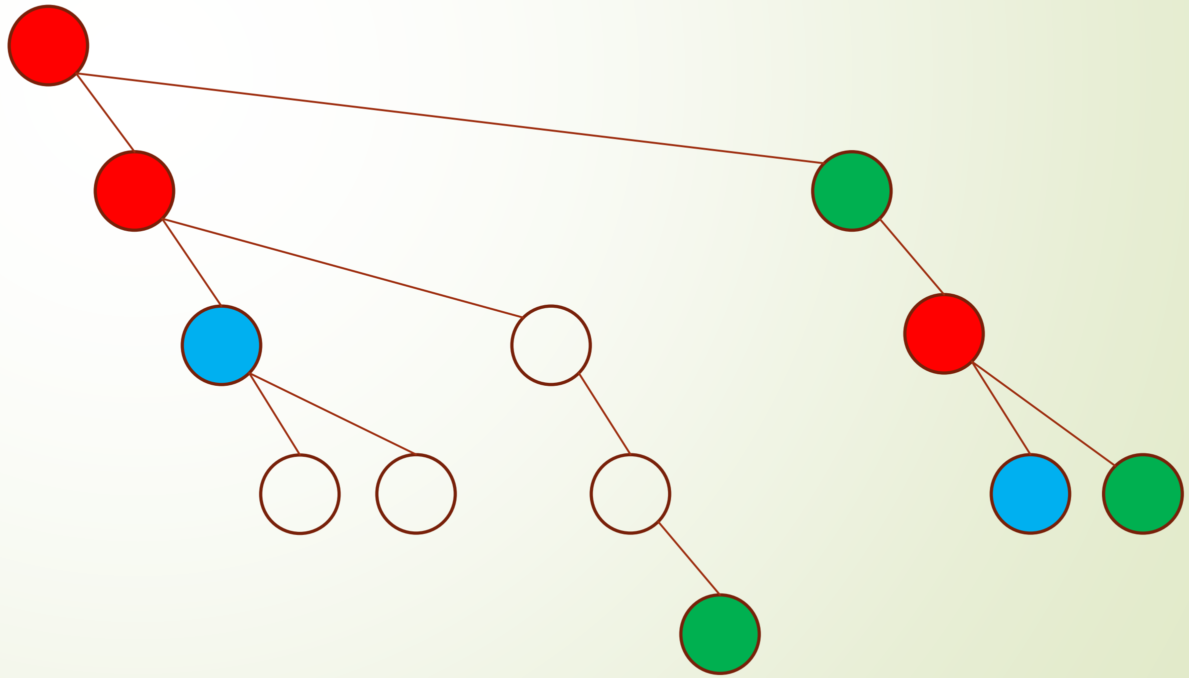
木の必要な部分を抽出

- まず，木に DFS 順序を与える



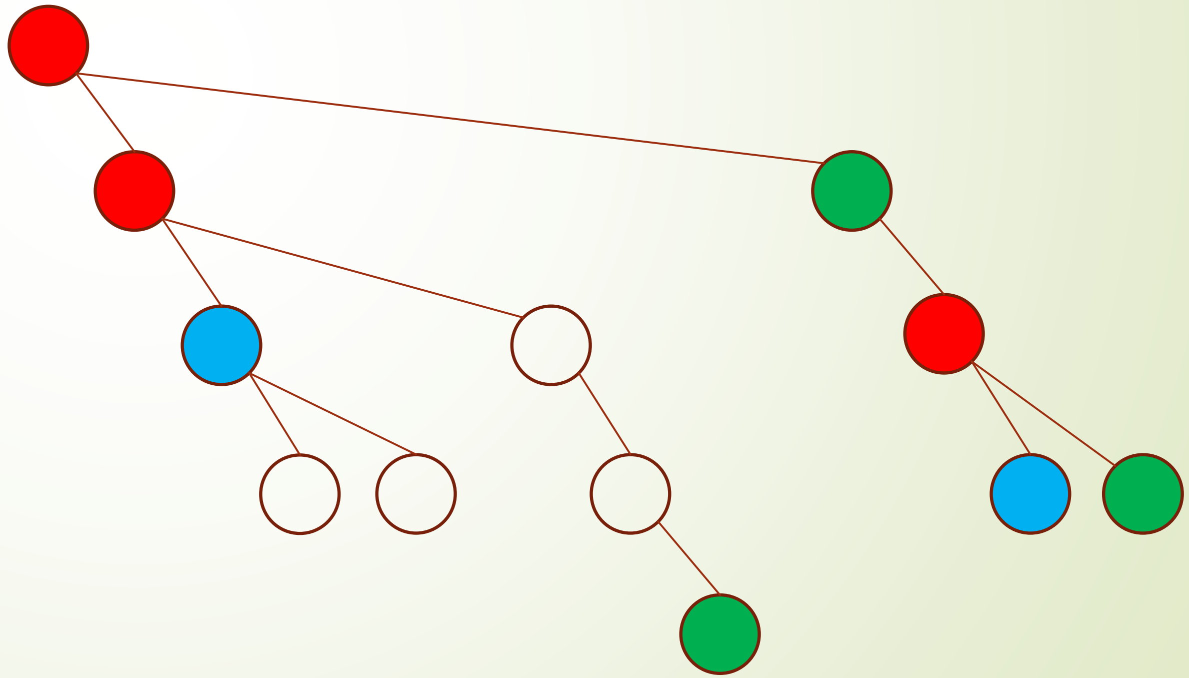
木の必要な部分を抽出

- 実は，この順序において隣り合う点同士の LCA だけ見れば十分



なぜか？

- この順序において「隣り合わない」点同士の LCA も，ある「隣り合う」点同士の LCA として現れることを言えばよい



なぜか？

- 2 点 P, Q の LCA を考える (R とする)
- R が P か Q と一致する場合は省略します
- 列の上で $P = S_1, \ldots, S_k = Q$ がこの順で隣り合っているとする
- R の子で，子孫に S_i を含むものを S_i' とする
- S_i' と S_{i+1}' が異なるなら， S_i' と S_{i+1}'の LCA は R (あるいはもっと上) になる
- 一方， S_i' は R の子孫であるから，このとき LCA は R になる
- LCA の定義から，P' と Q' は異なる

なぜか？

- LCA の定義から，P’= S_1’ と Q’= S_k’ は異なる
- よって， S_1’, …, S_k’がすべて一致するということはありえない

- ゆえに，DFS 列上で隣接する 2 点で，それらの LCA が R になるものが存在

点の数の評価

- 最初与えられる点の数は S + T 個
- 新たに必要になる点の数は，高々 S + T – 1 個
- よって，DP で必要になる点の数は全部で O(S+T) 個

- 新たな点を含めた木の構造がわかってしまえば，O(S+T) で DP が行える

「新しい木」の作り方の例

- まず，最初の点たちを X, Y 由来区別なく DFS 順序でソート
- 隣り合う点同士の LCA を求める
- RMQ などを用いて，LCA 列のある範囲でどこが最も根に近いか求められるようにする

「新しい木」の作り方の例

- 点たちのある範囲において木を構築する方法
- RMQ を用いて，その範囲でどこが根になるかを LCA 列の中で求める
- そのような点は複数回現れるかもしれないが，どこを選んでもよい

- その点の左側，右側について再帰的に木を構築する
- 左側，右側の根と，現在の根を結べば終わり
- 辺の長さは，元の木における「根からの距離」の差を用いる

- 同じ点が複数回現れるかもしれないが，結局長さ 0 の辺で結ばれるだけなので無問題

計算量

- 初期化
  - DFS 木の作成に O(N)
  - LCA を O(log N) で求めるための準備に O(N)
- クエリ (S, T の和をそれぞれ Ssum, Tsum とする)
  - DFS 順序に並び替えるのに O((S + T) log (S + T))
  - LCA をすべて求めるのに O(S + T)
  - LCA 列上での RMQ に，準備とクエリ合わせて O((S + T) log (S + T))
- 結局，O((N + Ssum + Tsum) log N)
- 満点が得られる