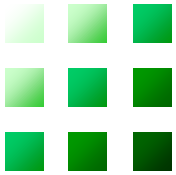


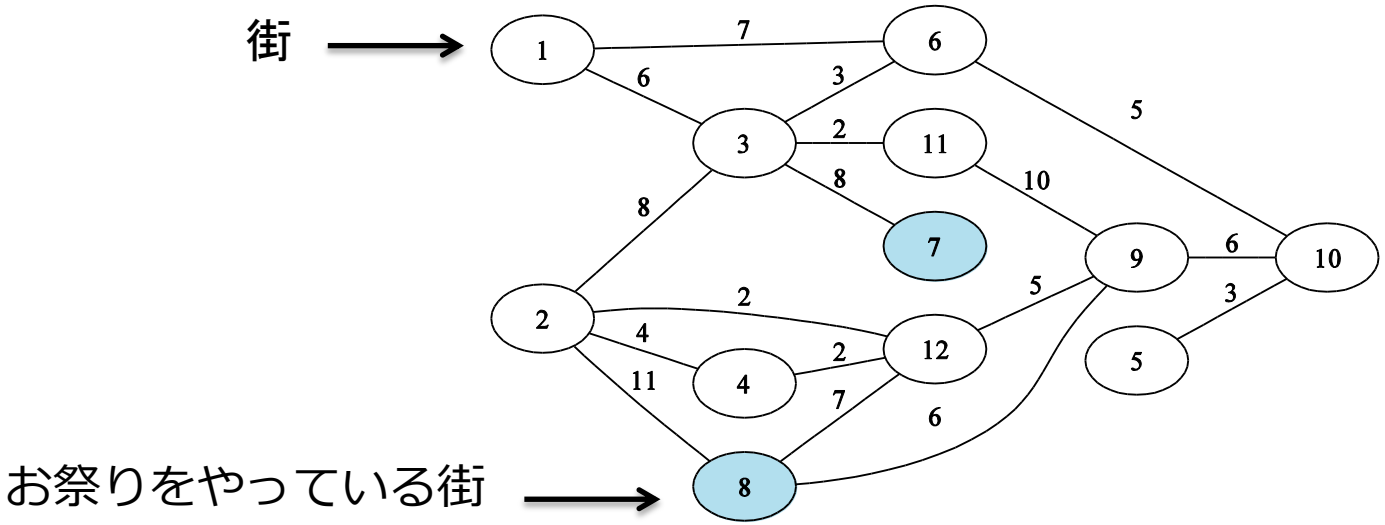
第 11 回 日本情報オリンピック本選 問題5

JOI 国のお祭り事情

秋葉 拓哉

（東京大学 情報理工学系研究科）

問題概要

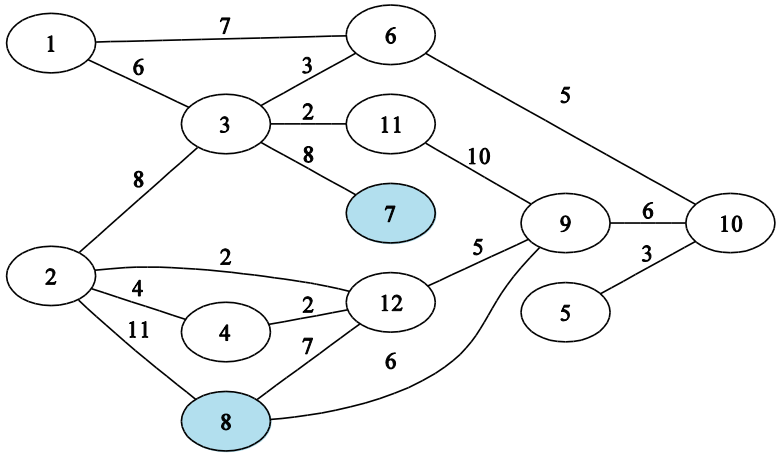


- できるだけお祭りに近づきたくない
- 街 s_i から街 t_i にどれだけ距離を保って移動できるか？
- …という質問に大量に答えたい（Q個）

用語など

グラフ

- 街 ＝ 頂点
- 道 ＝ 辺，枝



変数

N：街，M：道，Q：クエリ

スコア

この解説では，
各街の，お祭りまでの距離を，スコアと呼びます

できるだけスコアの小さな町を通らず移動したい

$O(Q\ M\ \log N)$ の解法 (10点)

10 点解法：ステップ 1

- まず，各街についてスコアを求める
  - スコア = お祭りまでの距離
- お祭りをしている街から **Dijkstra**
  - Dijkstra ：グラフで最短経路を求めるアルゴリズム
  - 今日は詳細は省略します（調べてください）
- 計算量は？
  - $O(M \log N)$

10 点解法：ステップ 2

- 各クエリ (s_i, t_i) に対し，最も良い経路を求める

- また，**Dijkstra** しましょう
  - 辺でなく頂点にコストがついている
  - コストは加算するのではなく最小を取る

- 計算量は？
  - 各クエリ $O(M \log N)$
  - クエリ Q 個に対し $O(QM \log N)$

別解

他にもいくつか解法があります

- Dijkstra を使う方法（今話した）
- Union-Find を使う方法（次に紹介）
- 2 分探索を使う方法（省略）

10 点別解

- 同様に各街についてお祭りまでの距離は求める
- 全道を，スコアの大きい順にソート
- 各クエリ (s_i, t_i) は以下のように処理
  - はじめ，全部の道は無かったと考える
  - 道を順に加えてゆく
  - 街 s_i と街 t_i が繋がったらやめる
  - 最後に加えた道のスコアが答え

10 点別解

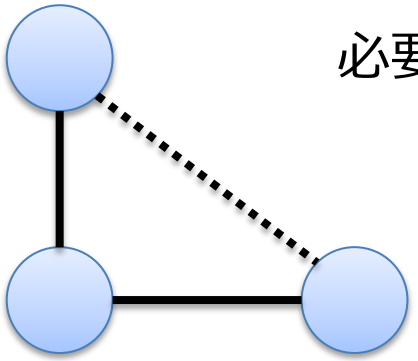
- 道を加えていき，繋がっているかを判定したい
- 道を加える度に深さ優先探索で判定？
  - 毎回$O(M)$時間かかってしまい非効率的
- Union-Find のデータ構造を使うと効率的
  - 毎回 $O(\alpha(N))$ 時間
  - 説明しません（ごめんね）
- 計算量：$O(M \log N + QM\alpha(N))$

$O(M \log N + Q N \alpha(N))$ の解法（30点）

30 点解法

- つなぐ 10 点解法をよく考えてみよう

- 既に繋がっているところを道でつなぐ意味なし
  - 特に新しいところに行けるようにならない
  - いらない



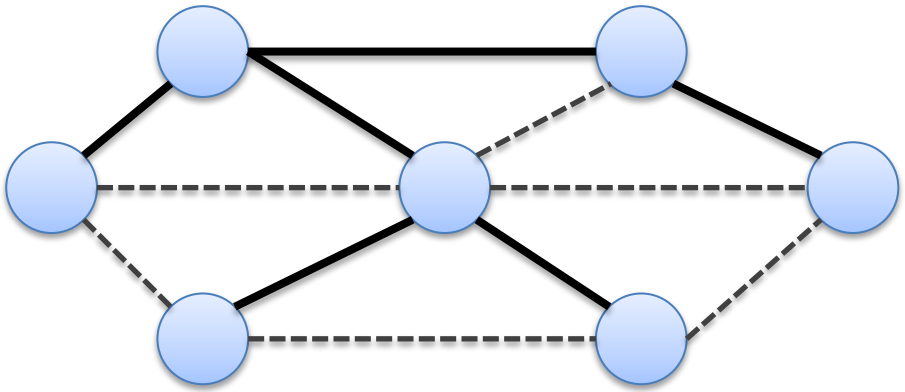
必要ない

もう繋がってる

30 点解法

- どの道が有用かをはじめに調べる
  - $O(M\ \alpha(N))$ でできる (Union-Find)
  - $N-1$ 本の道が有用として残る（全域木になるので）

- 各クエリに対しN 本の辺だけ見ればよい！
  - $O\big(QM\ \alpha(N)\big) \rightarrow O(QN\ \alpha(N))$

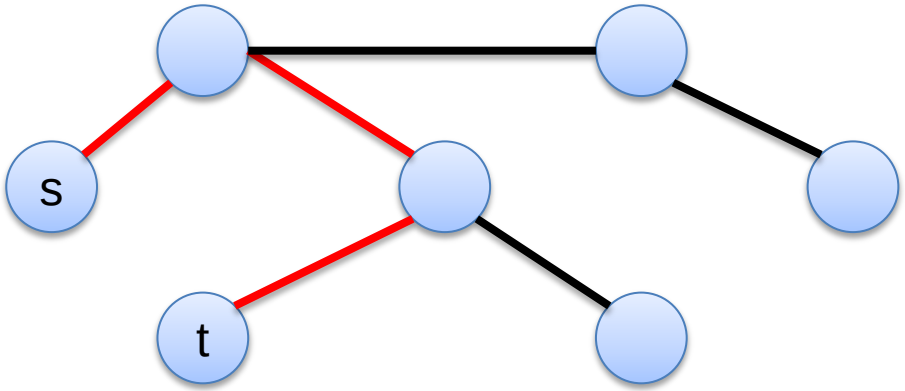


点線の辺は無かったことに
↓
道が M 本から $N-1$ 本に！

$O((M+Q)\log N)$ の解法 (100 点)

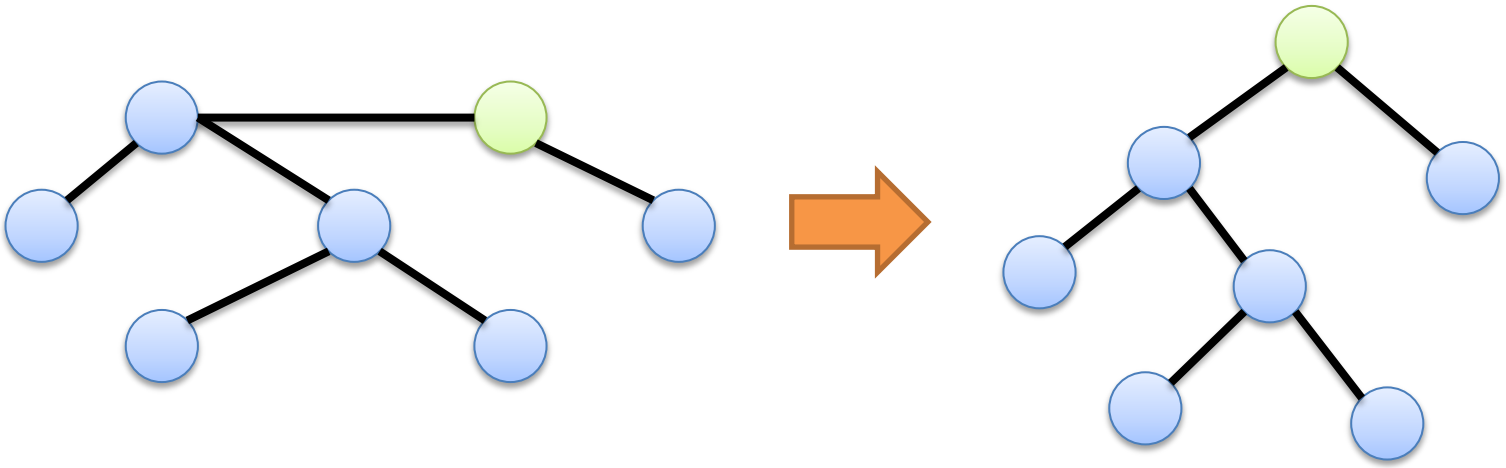
100 点解法

- つなぐ解法をさらによく考える
- 経路は，つないだ全域木の上での経路になる
- 木の中の経路に含まれる全ての街の中で，スコアの最小値が答え



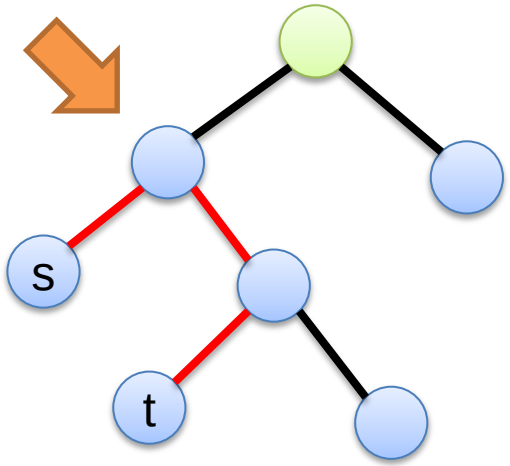
100 点解法

- 適当な頂点を根にして，根付き木にしよう



100 点解法

Lowest Common Ancestor



経路を 2 つに分けて考える
- s → LCA
- t → LCA

- Lowest Common Ancestor とその応用
  - LCA と，さらにそこまでの経路のスコアの最大値
  - ダブリング等を用いれば良い（省略）
  - 前計算 $O(N \log N)$ ，クエリ $O(\log N)$

100 点解法

- 少し工夫すると LCA だけで OK
  - 木を，頂点が上に行くほどスコアが低くなるようにうまく作る
  - LCA のスコアがそのまま答えに

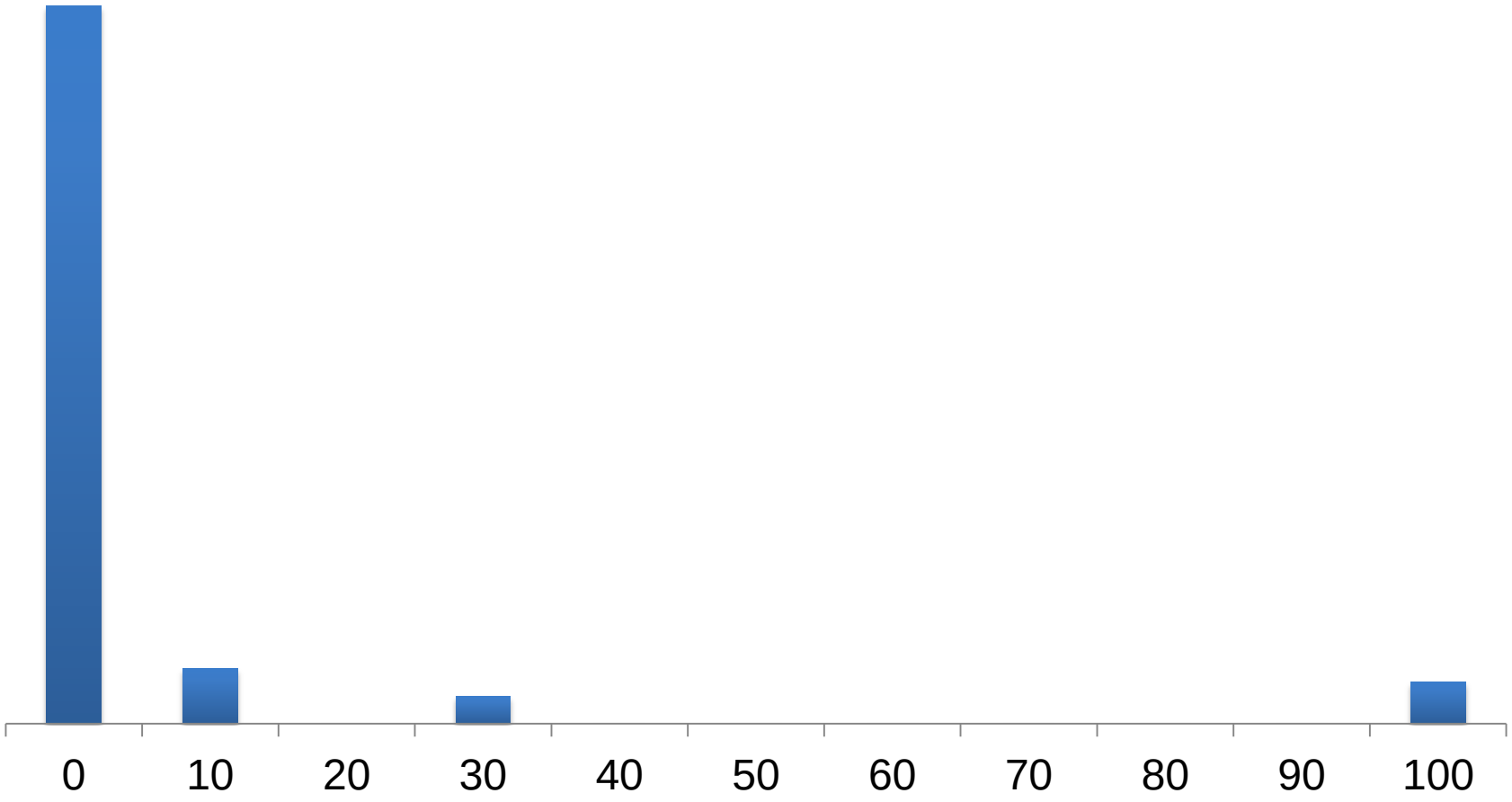
別解など

- クエリを頂点に集合として貯めこみ，マージしてゆく
  - マージ時にぶつかったら，そこがそのクエリの答え
  - $O(Q \log^2 N)$ になってしまうが多分間に合う，実装楽

- Union-Find を永続データ構造的に実装
  - 実際には縮約をなくせば親へのリンクに時間がつくだけ
  - つながる瞬間を調べたいので，時間で二分探索 $\rightarrow O(Q \log^2 N)$
  - （二分探索やめて LCA すると $O(\log \log n)$とかになってオモロイ）

- (おまけ：計算量だけなら)
  - 最短路は $O(M)$ のアルゴリズムあり
  - LCA は $O(N), O(1)$ のアルゴリズムあり
  - あわせて，$O(M+Q)$ で解ける，入力の線形＝最適！（実際は遅い）

点数分布



最後に

- 多くの重要なキーワードが登場しました
  - Dijkstra のアルゴリズム
  - 深さ優先探索
  - 二分探索
  - Union-Find のデータ構造
  - Lowest Common Ancestor

- 基礎的なものから発展的なものまで
  - 知ってれば解ける訳ではない
  - しかし，知らないと厳しい
  - ぜひ復習しておいてください