JOI 2013-2014 春合宿
Water Bottle 解説

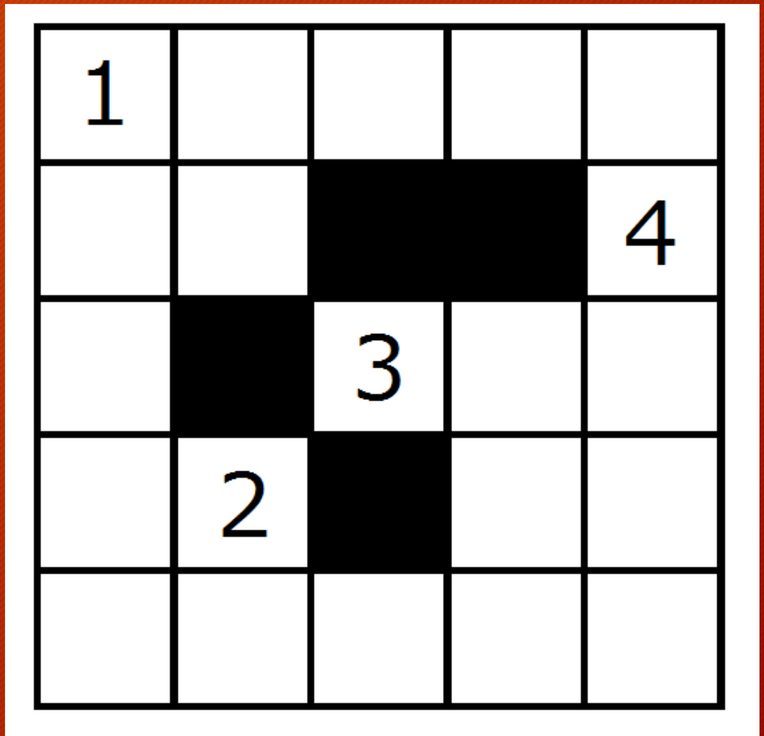
問題概要

- グリッド状の街があり，各マスは野原 or 建物 or 壁
- 壁のマスには入れない
- 野原を 1 マス通る時には水が 1 必要
- 建物では水を（水筒の容量の範囲内で）無限に補給できる
- 水は水筒に入れて持ち運ぶ

- 移動したい建物の組がいくつか与えられる
- それぞれの組について，必要な水筒の容量の最小値は？

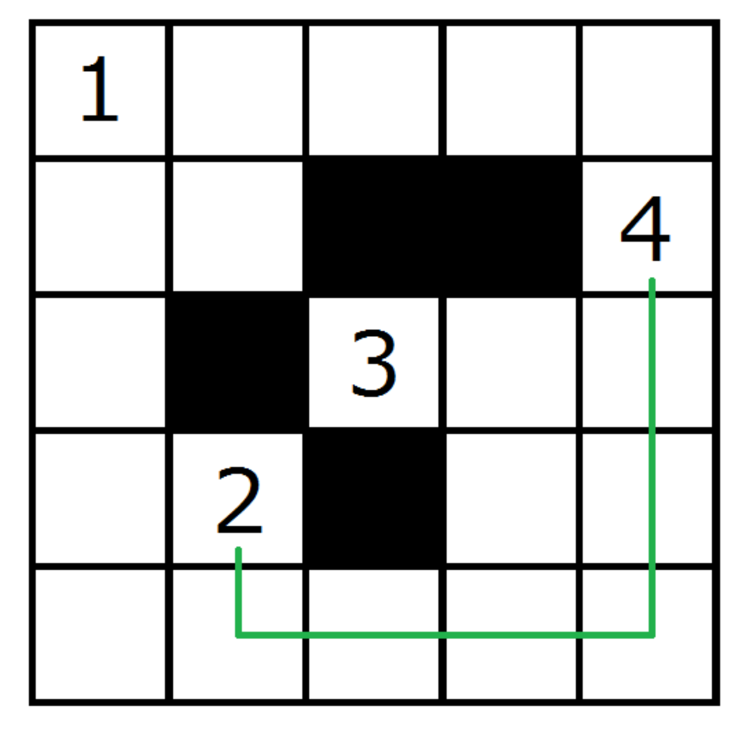
入力例（1）

- Sample 1



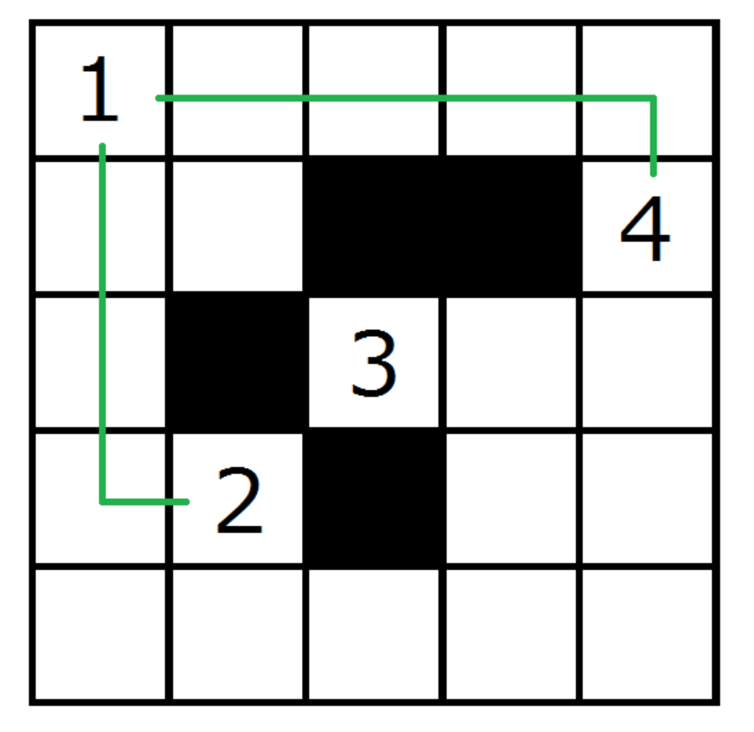
入力例（2）

- 建物　2 から建物 4 まで移動する例
  - 直接移動してみた
  - この例では野原を連続して 6 マス通るので，水筒の大きさは 6 必要



入力例（3）

- 建物 2 から建物 4 まで移動する例
  - 遠回りだけど建物 1 を経由してみた
  - 2 -> 1 は野原 3 マス
  - 1 -> 4 は野原 4 マス
  - 水筒の大きさは 4 で十分
- 水筒の大きさを最小化するのが目的
- 本当の最短路には興味がない



小課題 1（1）

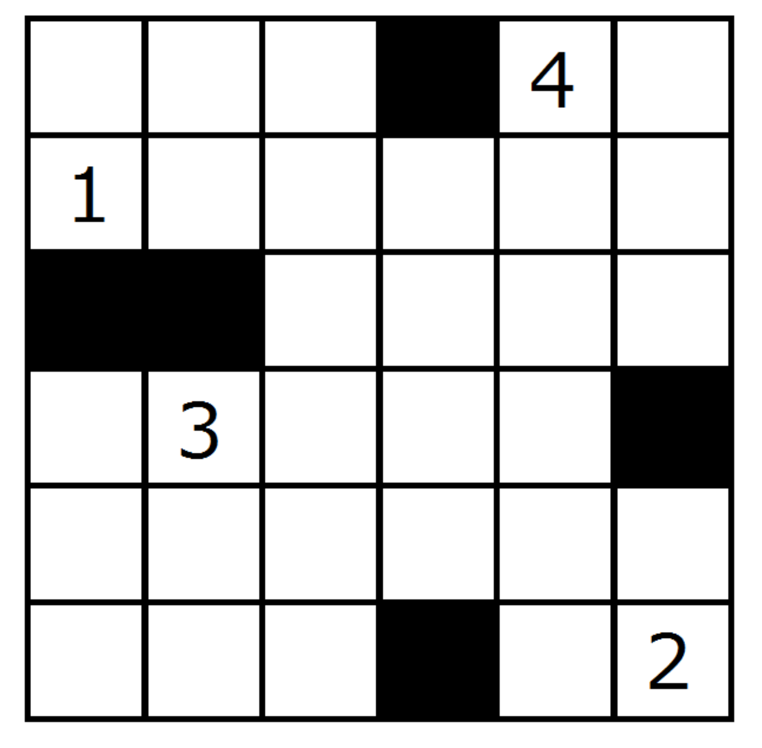
- 建物と建物の間を直接（途中で水を水筒に追加せずに）移動することをまず考える
- ある移動について，必要な水の量は「グリッド上の最短距離 – 1」
- 各建物から BFS をすれば，すべての最短距離は O(HWP) で求められる

小課題 1（2）

- ある移動を考えたとき，必要な水筒の大きさは？
  - 明らかに，その移動中の建物間の直接の移動の中での，必要な水の量の最大値
- 最短路を求めるような気分で，「パス上重みの最大値」の最小値も求められる
- P 個の頂点について，全点間についてこの値は Warshall-Floyd で O(P^3) で求められる（するとクエリはこの値を呼び出すだけ）
- 全部で O(HWP + P^3 + Q)
  - 小課題 1 が解けて，10 点が得られる

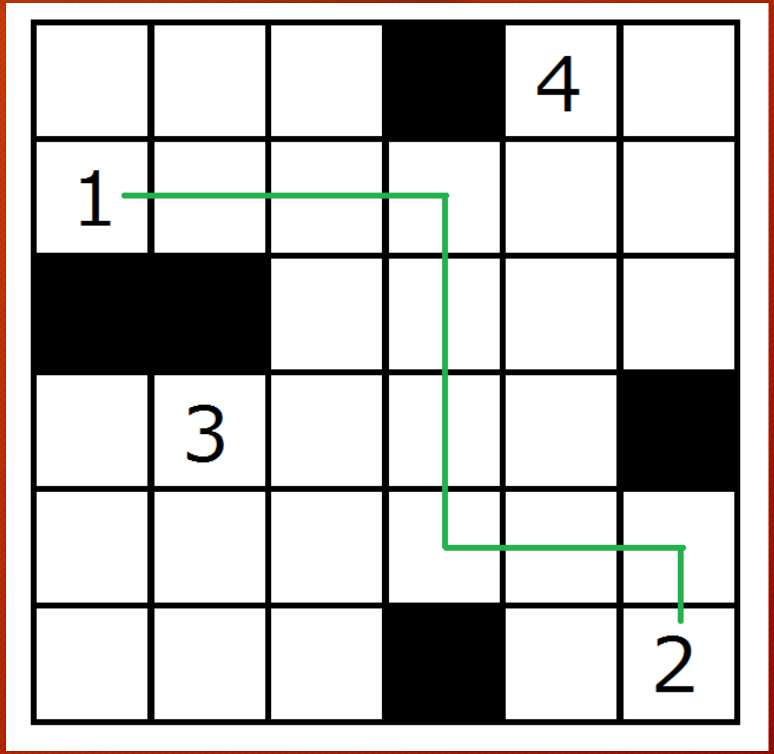
満点解法のために(1)

- 意味のある「直接の移動」とは何か？
- 右の図で 1 から 2 まで移動することを考えてみる



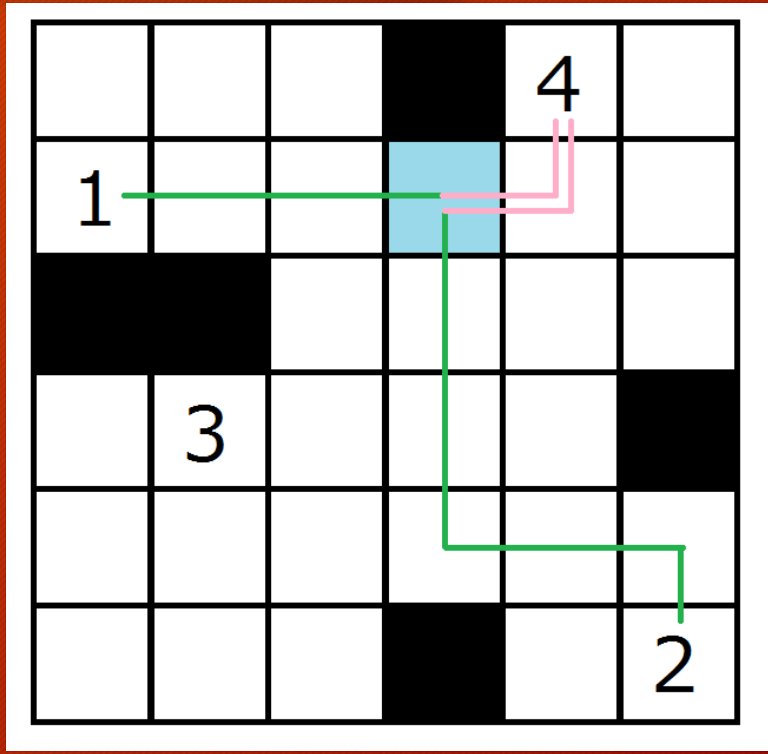
満点解法のために(2)

- とりあえず適当に直接の経路を書いてみた
- まだ無駄がありそう



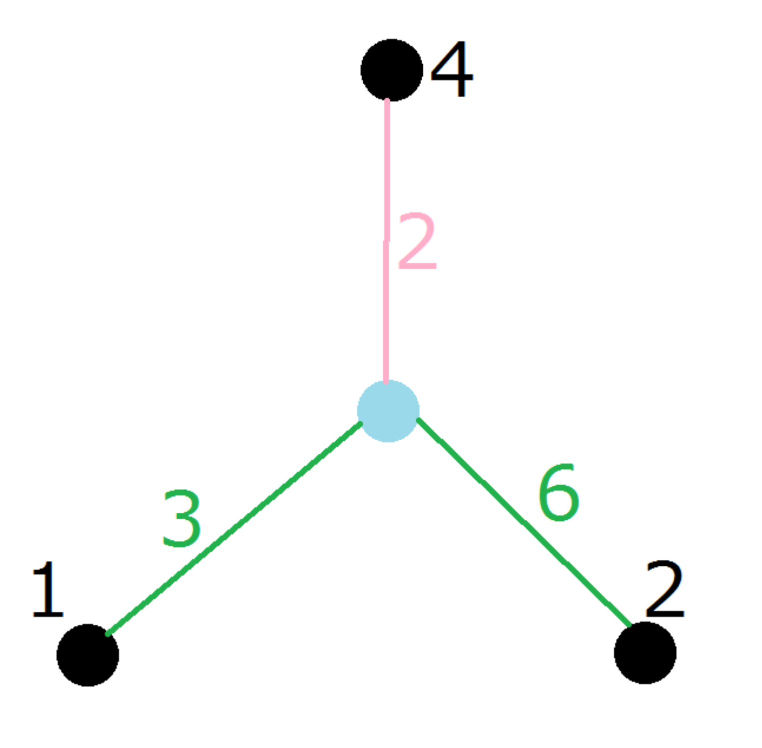
満点解法のために(3)

- 例えば右のように，途中で 4 に寄るともっと水筒の容量が少なくてよくなる(これでも明らかに無駄があるけど)
- 最初の移動で水色のマスを通っていたが，そこからあえて 4 に寄り道するようにした
- そうしたら，必要な水筒の大きさが小さくなった



満点解法のために(4)

- 関係するマスたちの距離についての構造のみ取り出してみた
- 1-2 を直接移動すると，最短距離は 9
- 1-4，4-2 と移動すると，それぞれ最短距離は 5，8
- 水色マスが 1，2 よりも 4 に一番近いので，4 に寄り道したほうがうれしい

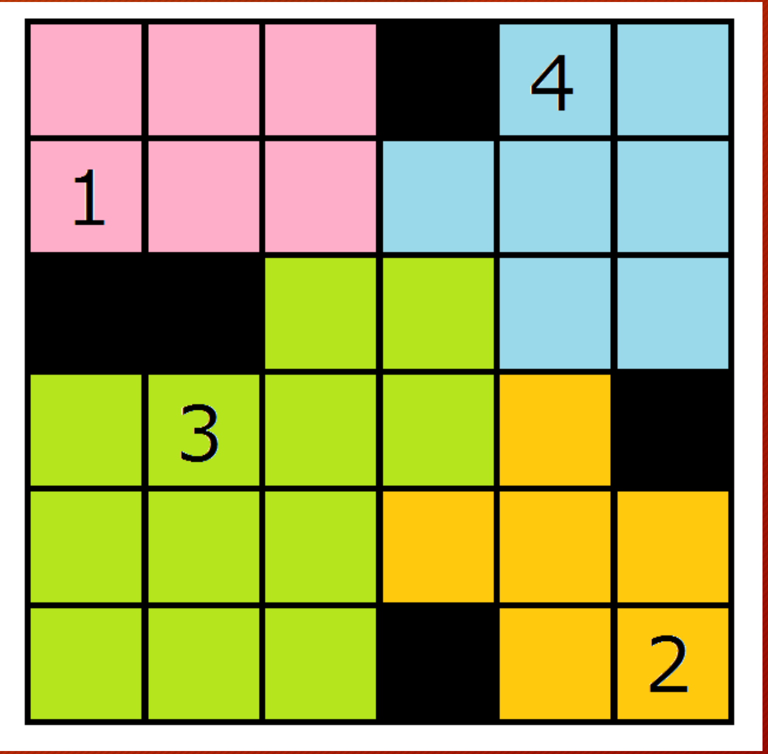


満点解法のために(5)

- 「a から b まで」直接移動しているときに，最寄りの建物が別の c になったら c を経由するとしてよい
- 逆に，直接移動するときは，常に最寄りの建物が a か b であるべき
- 実際には境界の問題もあるが，「他の建物までの距離も同じ」場合でも寄り道をして困ることはない

満点解法のために(6)

- 最寄りの建物で色分けすると右のようになる
- 建物間の移動では，色の境界を 1 回以上またぐ
- 逆に色の境界について考えると，その両端のマスからはそれぞれの最寄りのマスへ直行するのが最適
- 色の境界は O(HW) 個なので，意味のある移動も実は O(HW) 個しか存在しない！



小課題 2

- 各マスの最寄りの建物を求めるのは，各建物から BFS をすれば O(HW)
- さらに O(HW) で，意味のある移動を全部列挙できる
- 重み最大値が最小のパスを求めるのは，Dijkstra が使える
- 密グラフについての Dijkstra を用いると，クエリあたり O(P^2)

- 全部で O(HW + P^2 Q)
  - 小課題 2 が解けて，30 点が得られる
  - これだけでは小課題 1 は解けないことに注意
- 必要な辺の数が多いので，Dijkstra を O(E log P) でやっても小課題 3 を解くのは難しい

小課題 3（1）

- クエリをもう少し高速化したい
- 水筒の大きさがどれだけあれば移動が可能か？と考える
  - 必要な大きさが小さい「直接の移動」から順にできることにしていって，いつ移動ができるようになるか調べる

小課題 3（2）

- 移動可能性は推移的
  - （x，y）の間が移動でき，（y，z）の間が移動できるなら（x，z）の間も移動できる
- 建物たちをグループに分けて，同じグループの間は移動できる，とする
  - 新たに移動できるようになった場所は，グループを併合する
- このような処理は Union Find でできる

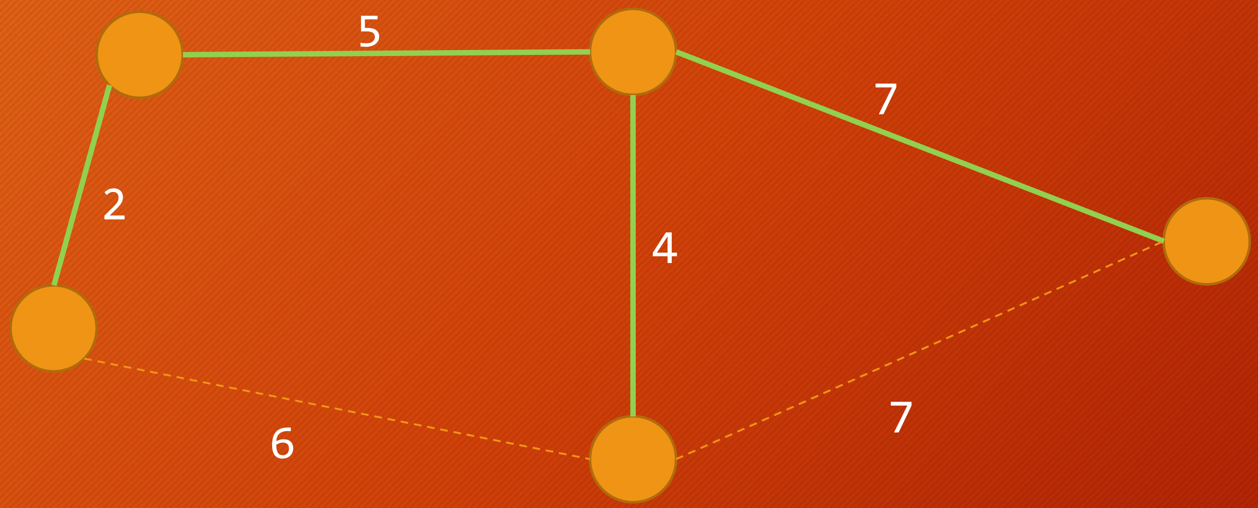
小課題 3（3）

- 距離の小さい「直接の移動」から見ていって，順に Union Find でつなげる
- このとき，すでにつながっている 2 点間を結ぶ移動は使う必要がない！
  - 距離がそれ以下の「直接の移動」を用いて移動することができる
- つながっていない 2 点間を結ぶのは，高々 P-1 回
- 結局，考えるべき「直接の移動」は高々 P-1 個で済む

- 最後までつながらなかった点たちは，十分大きい INF を距離としてつなげておくとよいです

小課題 3（4）

- 下のグラフにおいて，緑色の辺以外の辺はないと思っても問題ない



小課題 3（5）

- 結局これは Kruskal 法と全く同じです
  - 最小全域木とは目標が少し違うが，同じ結果になる
- 得られるものは木になる
- 最短路は，普通に DFS とかで求めても O(P) で求められる

- 結局，O(HW + PQ)
  - 小課題 1, 2, 3 が解けて，70 点が得られる

小課題 4

- 木の上で，パス上最大値を求めるクエリを高速で処理したい
- 木を勝手に根付き木にする
- Doubling の手法を使って LCA を行う
  - 最大値も同時に前計算させて同時に求める
  - 前処理 O(P log P)，クエリ O(log P)
  - 「蟻本」を見ましょう

- 全部で O(HW + (P + Q) log P)
  - 満点が得られる

別解（1）

- 作る木が，必ずしも実際の構造に即している必要はない
- 「小さい方から見ていった時の連結性」さえ一致していれば何でもいい

別解（2）

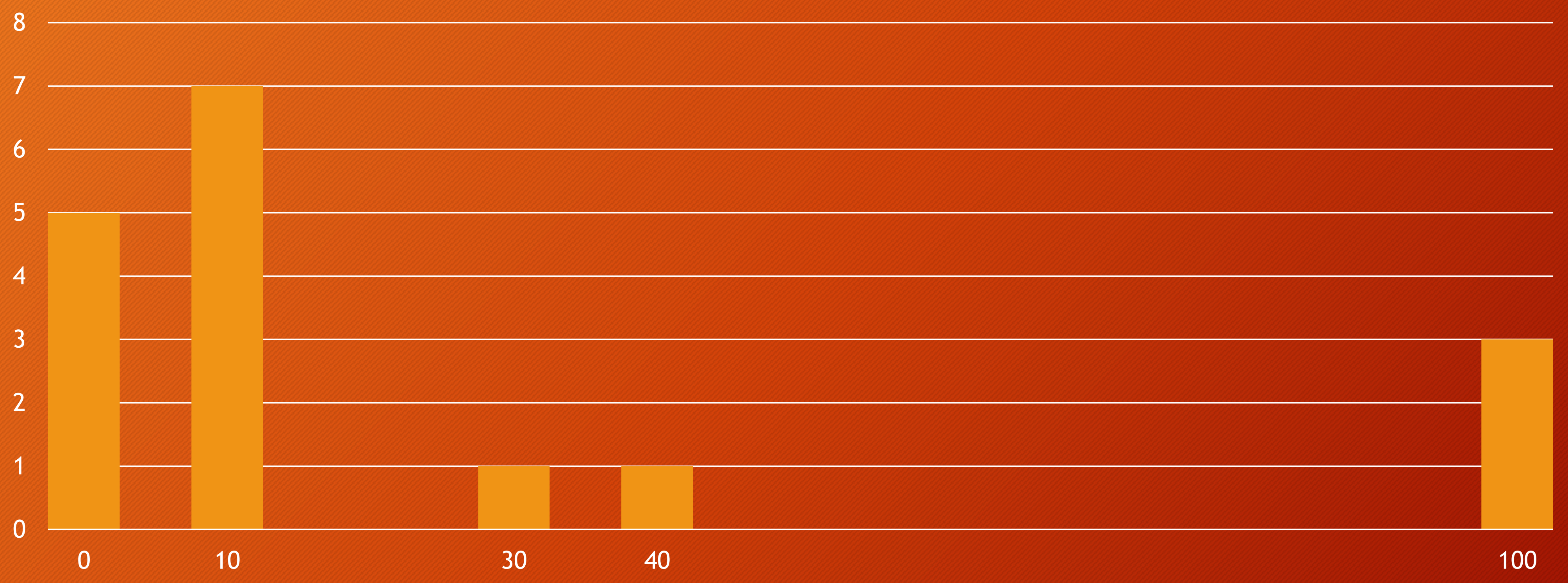
- 下のような変形をしても問題ない

別解（3）

- Union Find 木で，経路圧縮をせずに，辺に「どの距離の移動によってつながったか」を覚えさせる
- すると，2 点間のパス上の最大値が答えになる
- 木は，「小さい方を大きい方の下につける」方法により高さを O(log P) にできる
- すると，根付きにして「共通の点になるまで 1 段ずつ根に近づく」方法をやるだけでクエリあたり O(log P)

- 結局 O(HW + (P + Q) log P) なので，これも満点が得られる
  - さらに LCA で Doubling をしたりするとクエリあたり O(log log P) にまでできる

得点分布



得点分布