招待 (Invitation)

JOI 春合宿 2012 Day 4

解説： 保坂 和宏

問題概要

- A 匹の犬と B 匹の猫
- N 個の仲良しグループ
  - 犬 P_i から犬 Q_i まで，猫 R_i から 猫 S_i まで
  - 仲良し度 T_i

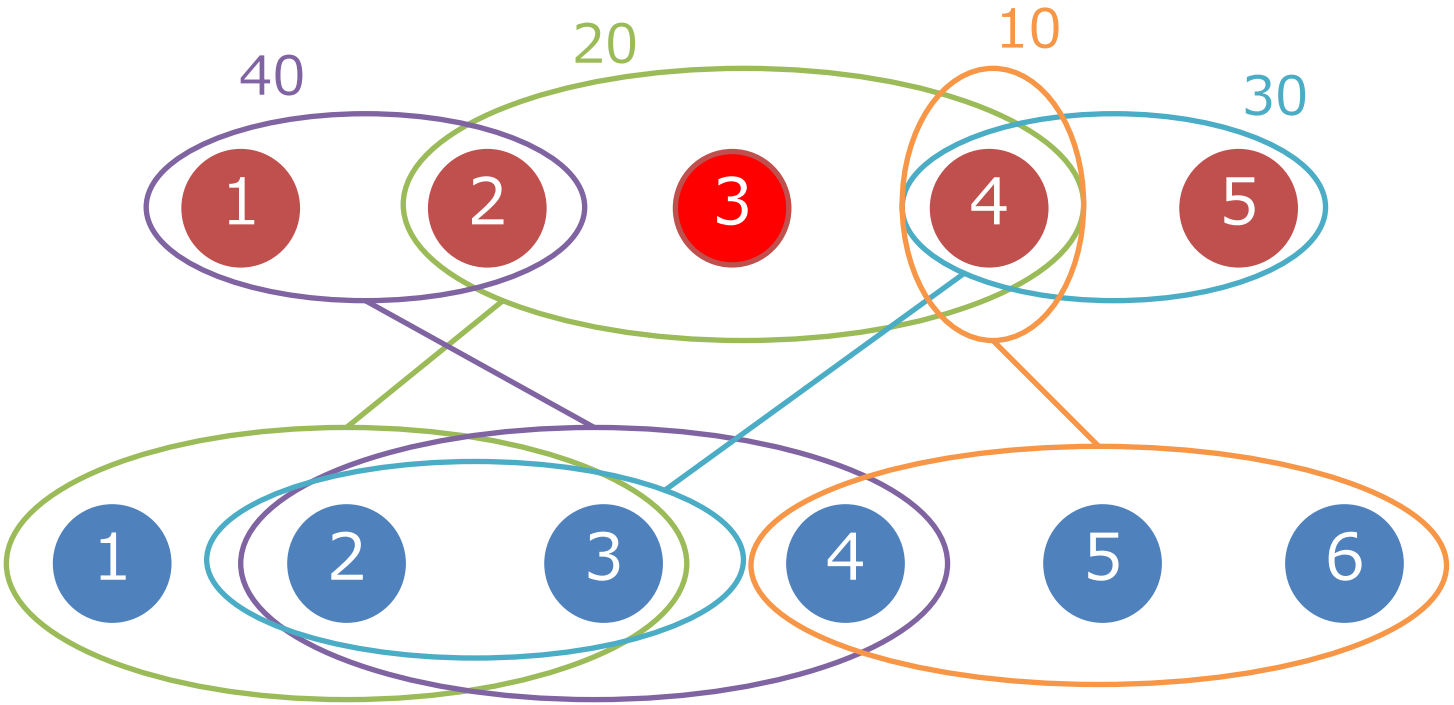
- みんなをパーティーに招待する
  - 問題文に招待のアルゴリズムが書かれている

問題概要

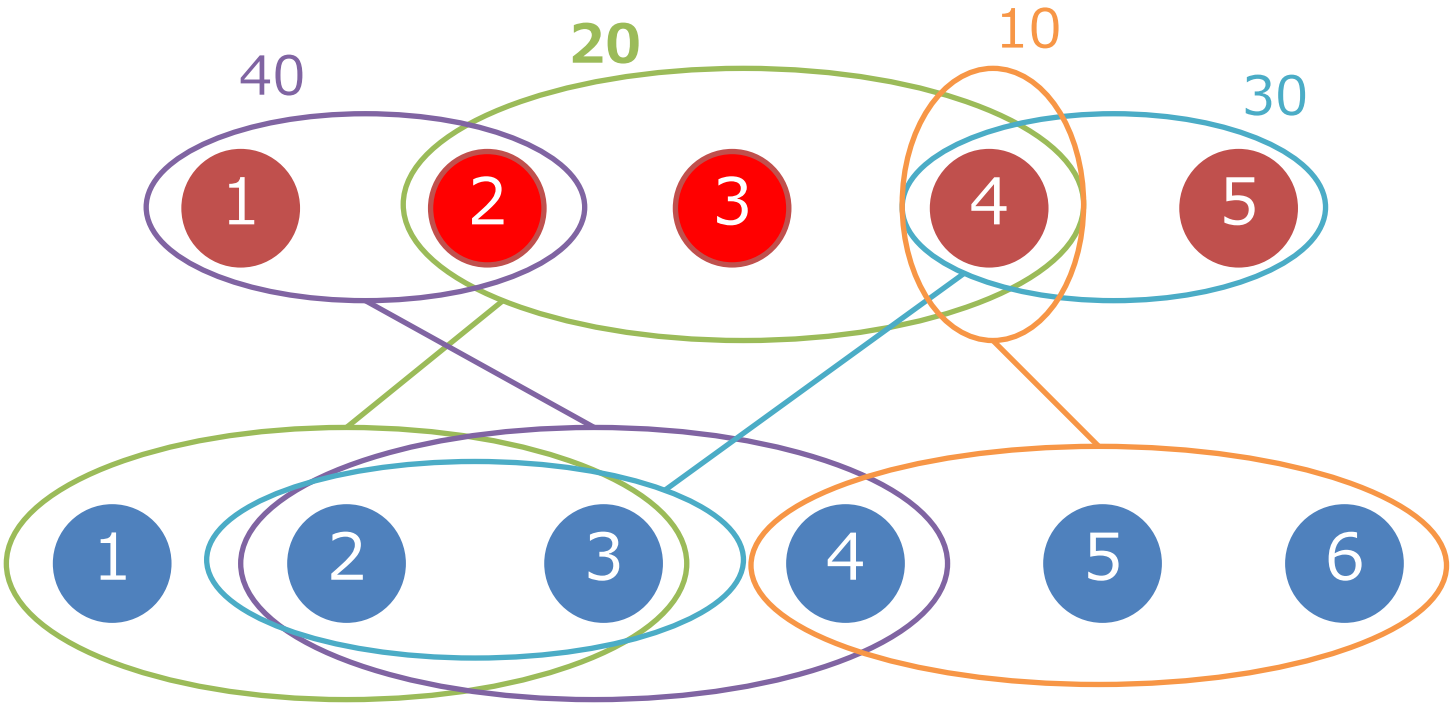
1. 犬 C を招待
2. 幸せ値最大の動物を招待することを繰り返す
   - 幸せ値：既に招待した動物と一緒に所属する仲良しグループのうちの仲良し度の最大値
   - 同じときは犬，番号小さい方を優先
   - 幸せ値 0 の動物しかいないときは失敗

- 幸せ値の和を答える

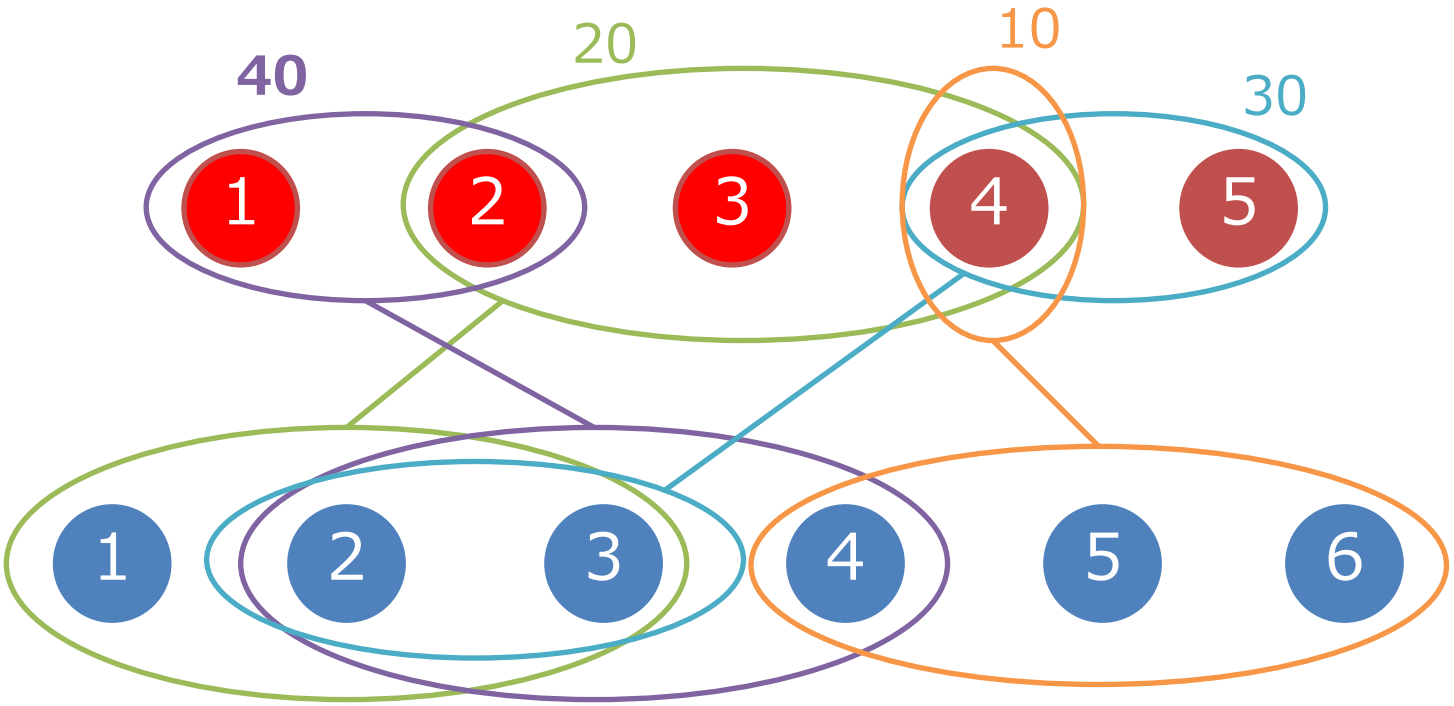
問題概要



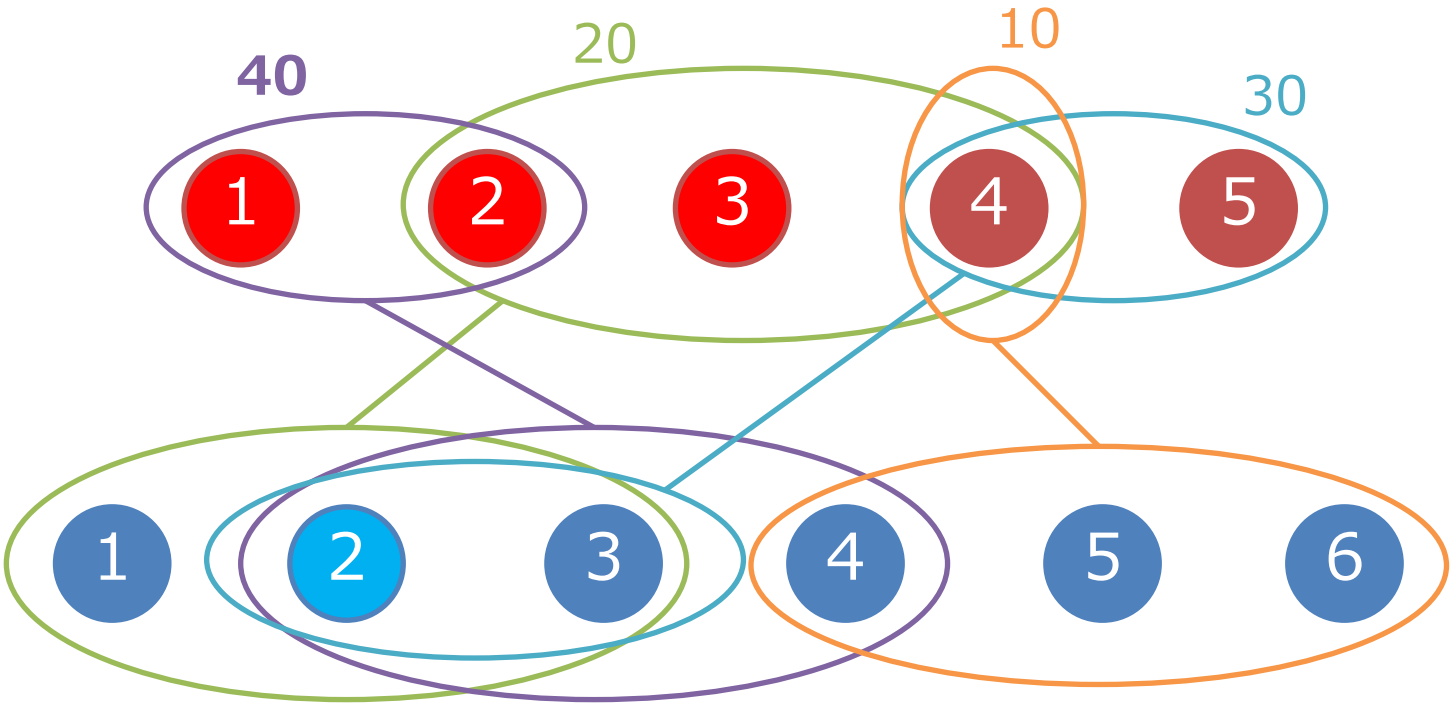
問題概要



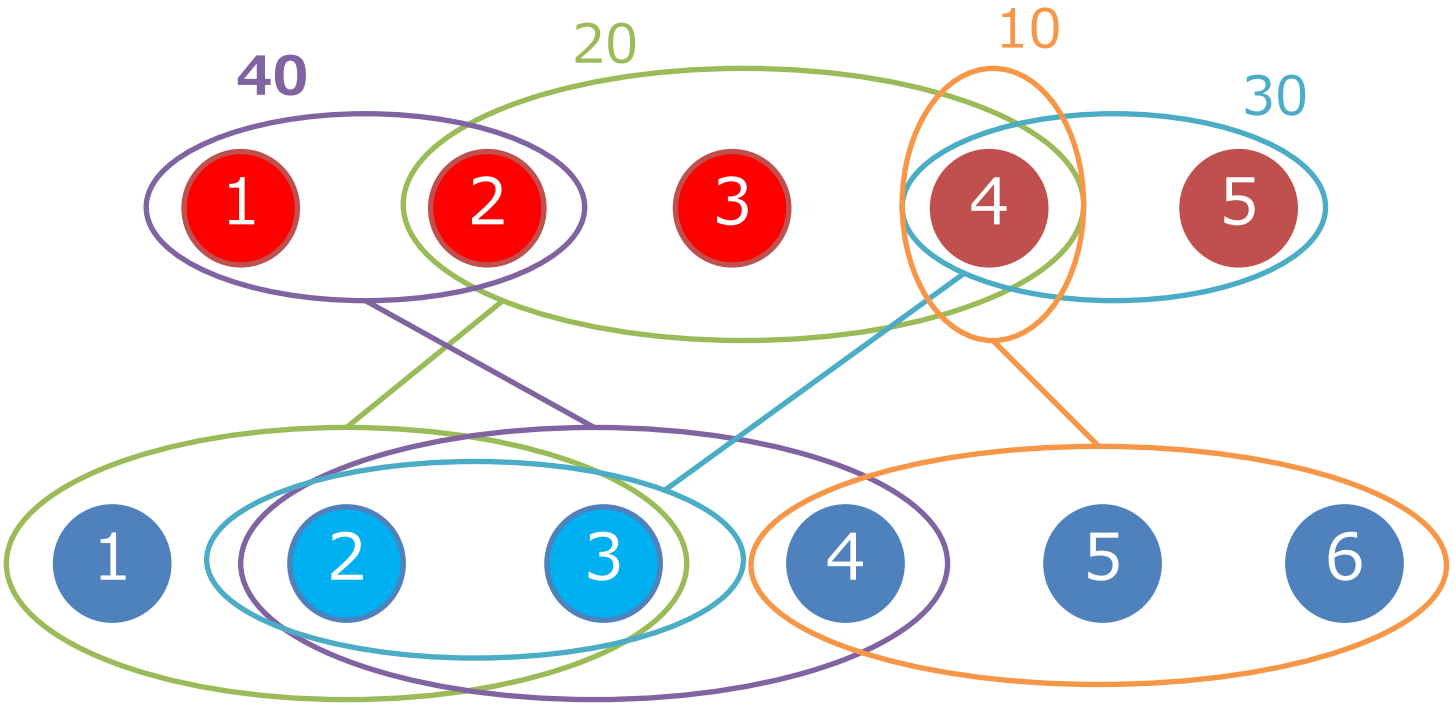
問題概要



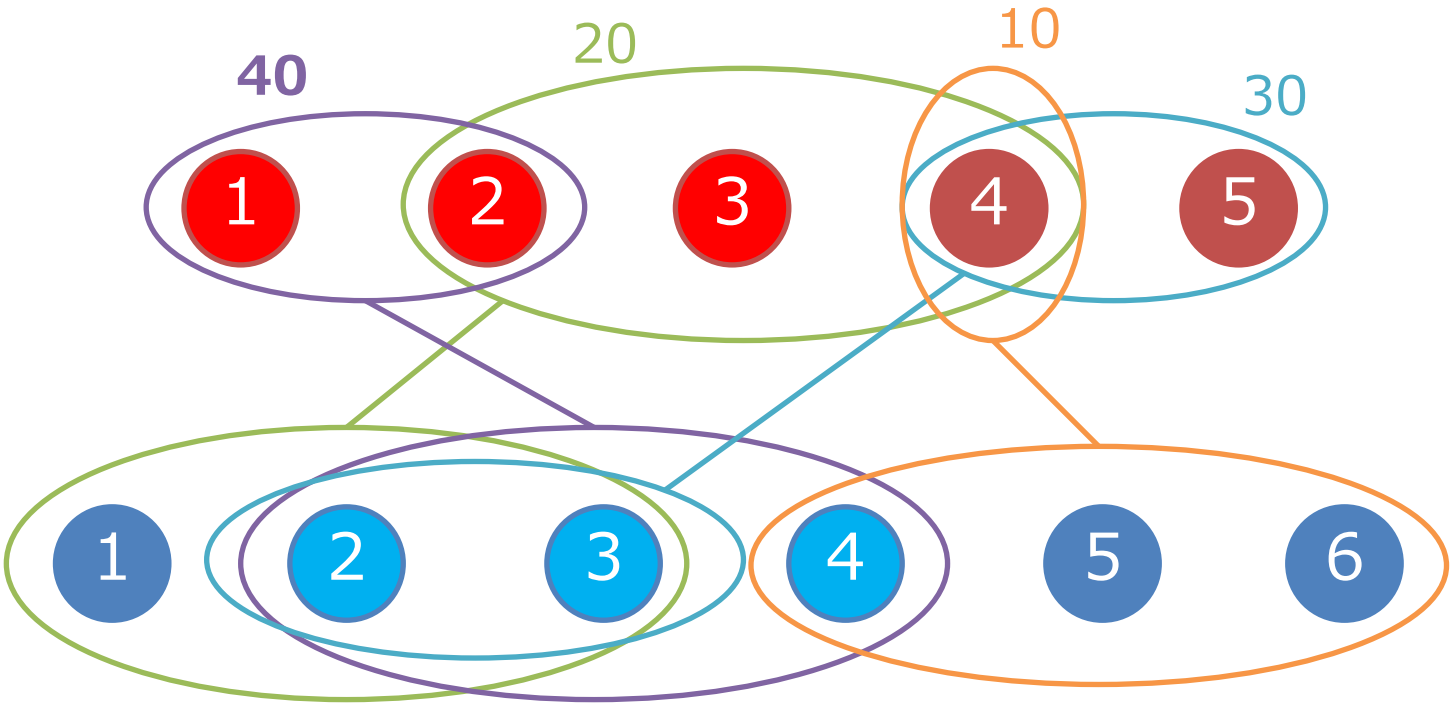
問題概要



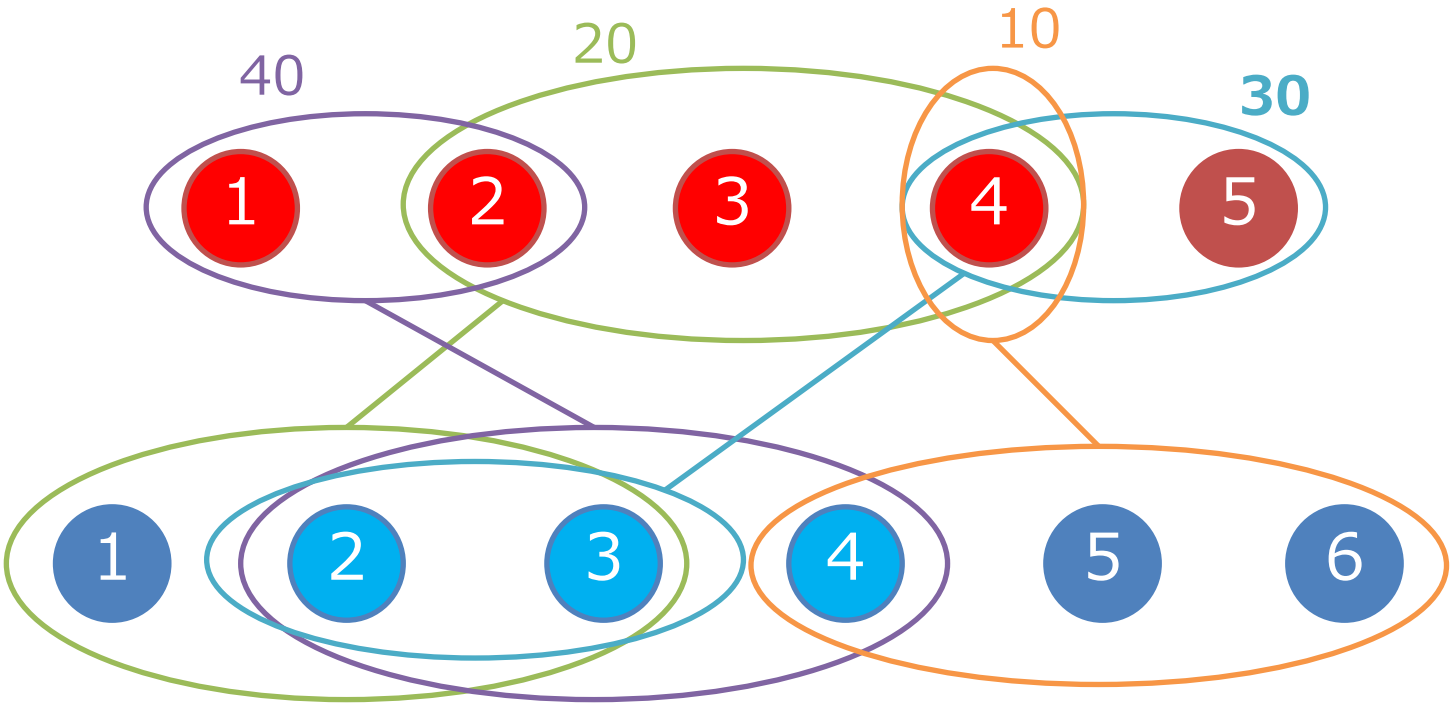
問題概要



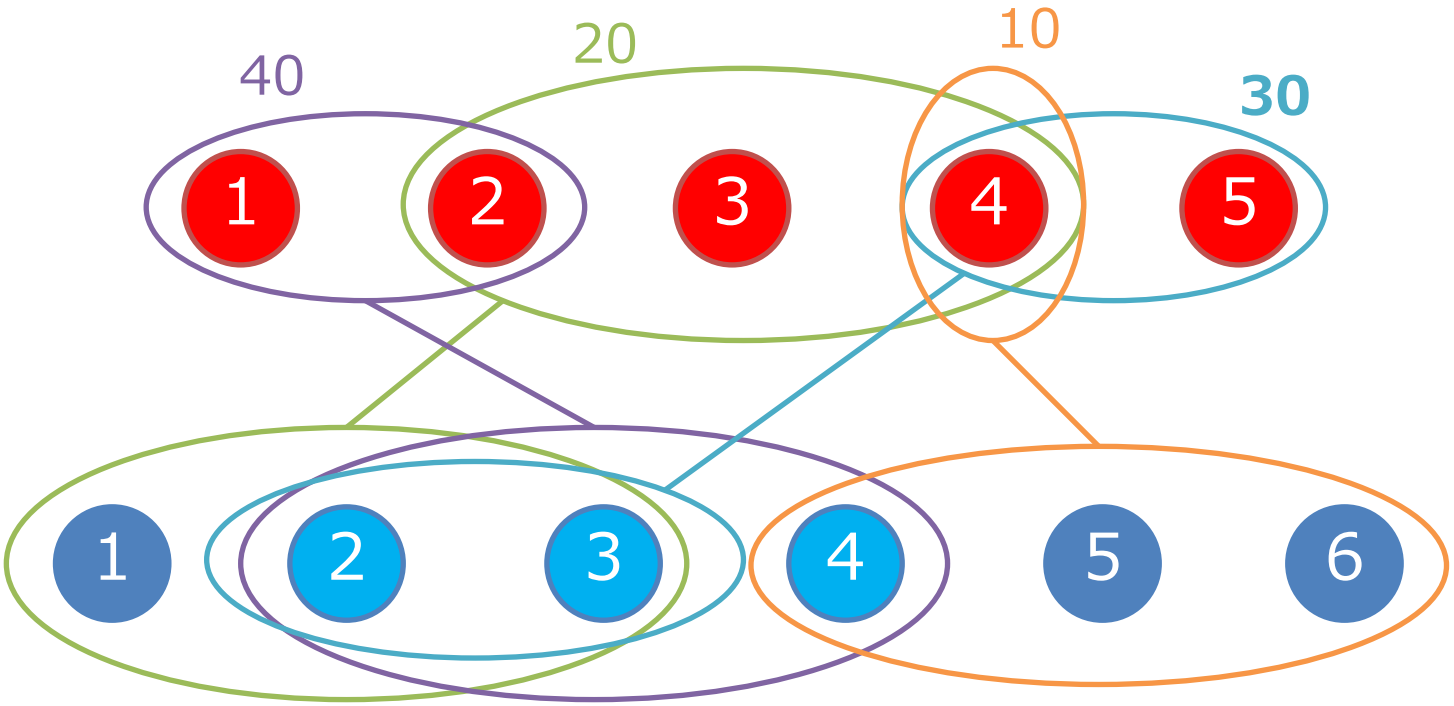
問題概要



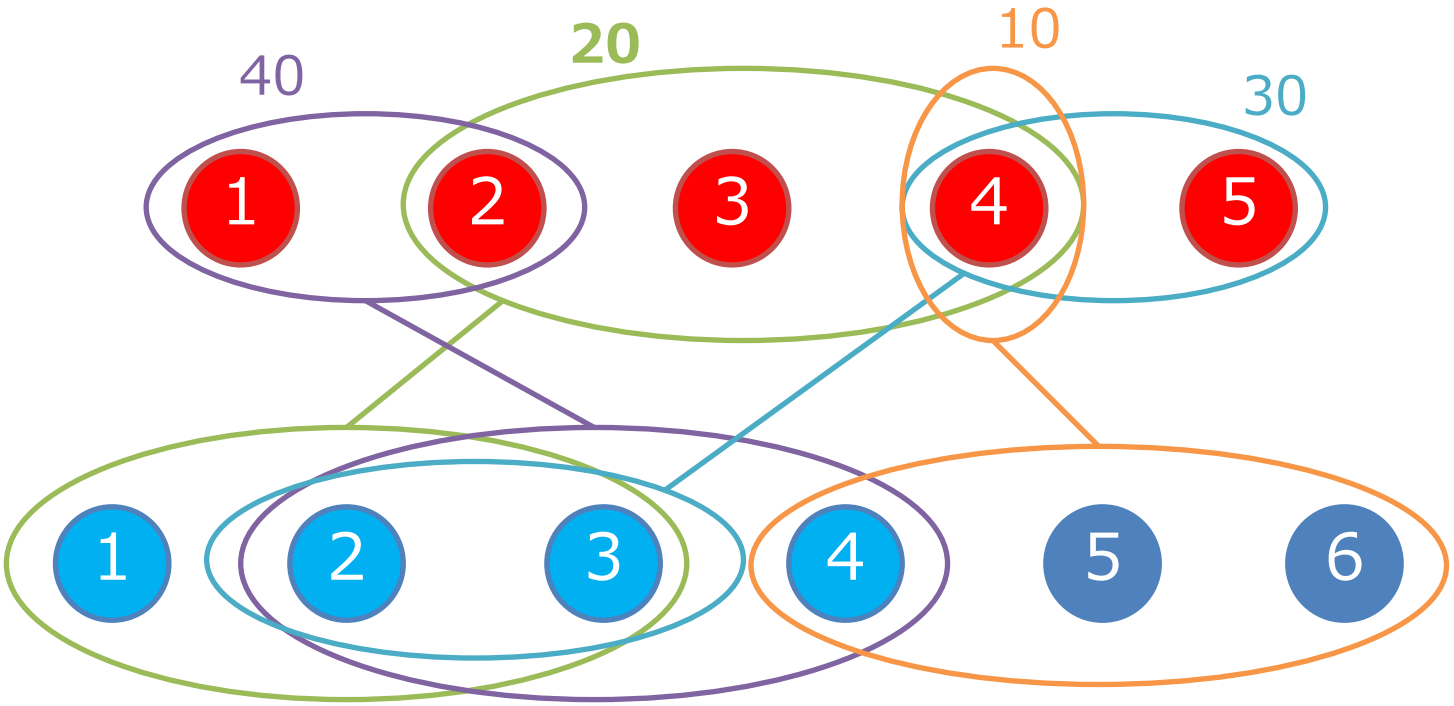
問題概要



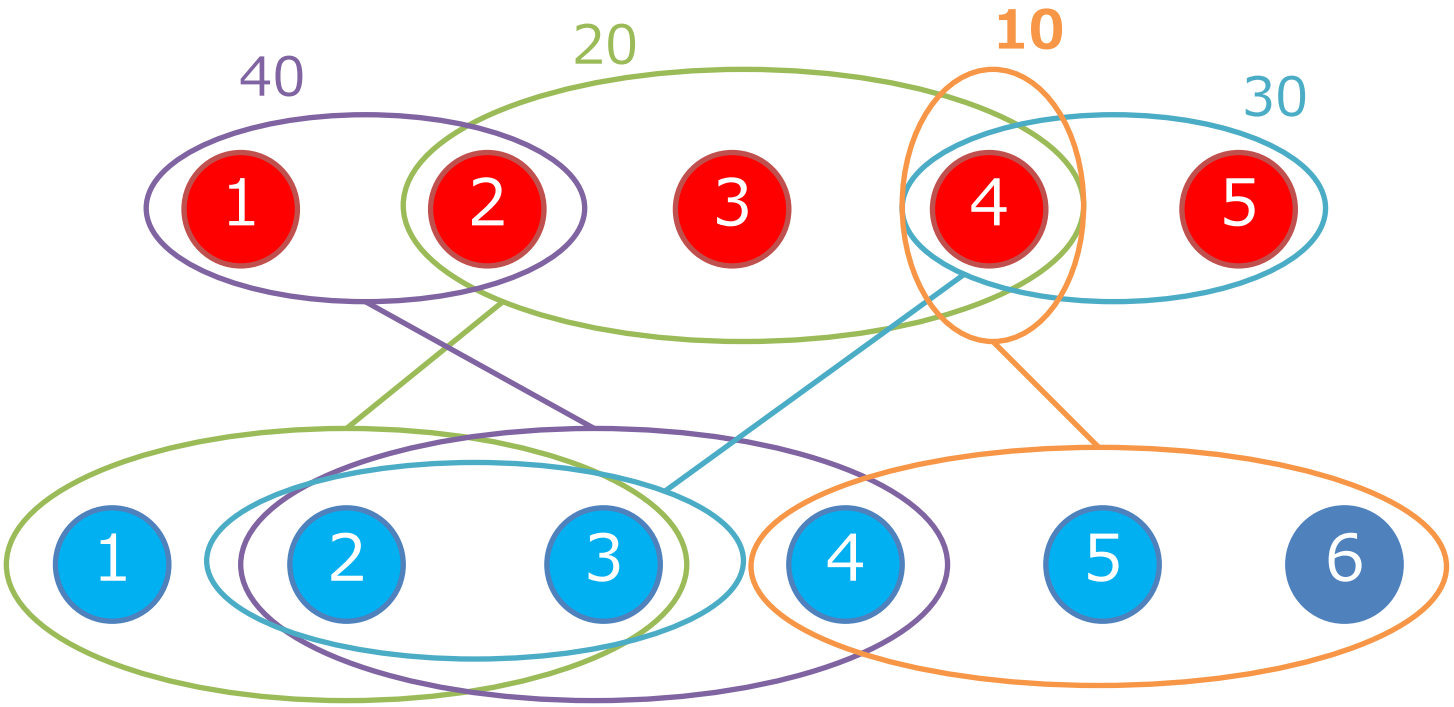
問題概要



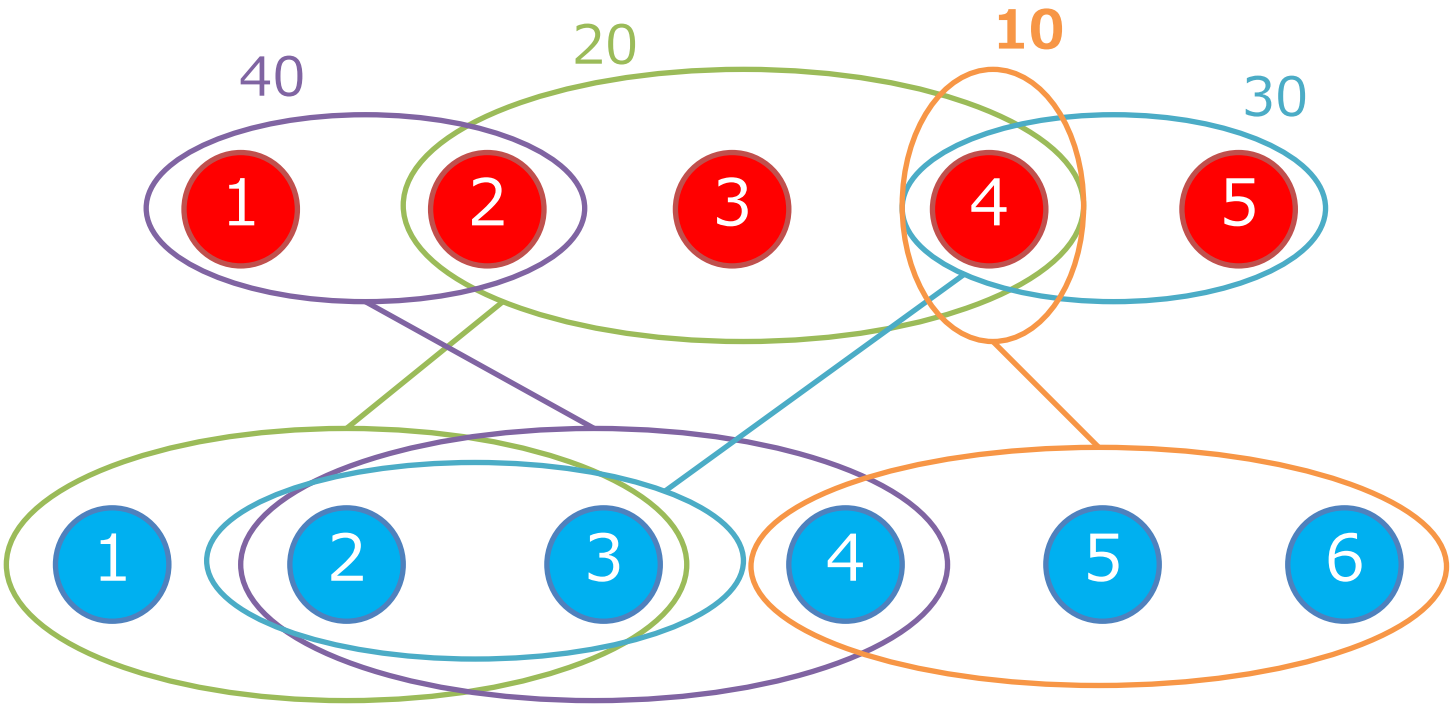
問題概要



問題概要



問題概要



制約

- $N \leqq 100{,}000$
- $A, B \leqq 1{,}000{,}000{,}000$

- 50 点 … $N \leqq 2{,}000$
- 30 点 … $N \leqq 2{,}000$, $A, B \leqq 1{,}000$

シミュレーション解法

- 問題文に書かれた通りにシミュレーションする
  - 実装の指針はいろいろ
  - 繰り返しは $A + B$ 回，$O(A + B)$ 匹のうち最大を求める
- $O(N (A + B)^3)$ or $O(N (A + B)^2)$
  - 0 点 ～ 10 点？

シミュレーション解法

- シミュレーションを効率よくしたい

- 誘われていない動物の幸せ値は減らない
- 幸せ値が更新されるのは高々 N 回

シミュレーション解法

- $A + B$ 回の繰り返しに対して,
  - 幸せ値最大を選ぶ ($O(A + B)$)
  - 各仲良しグループに対して ($O(N)$),
    - 新しい仲良しグループなら，幸せ度を更新する ($O(A + B)$)

- $O(N(A + B) + (A + B)^2)$
  - 30 点

シミュレーション解法

- 制約は A, B の値が大きい
- 仲良しグループへの入り方が同じな犬 or 猫は番号が連続している

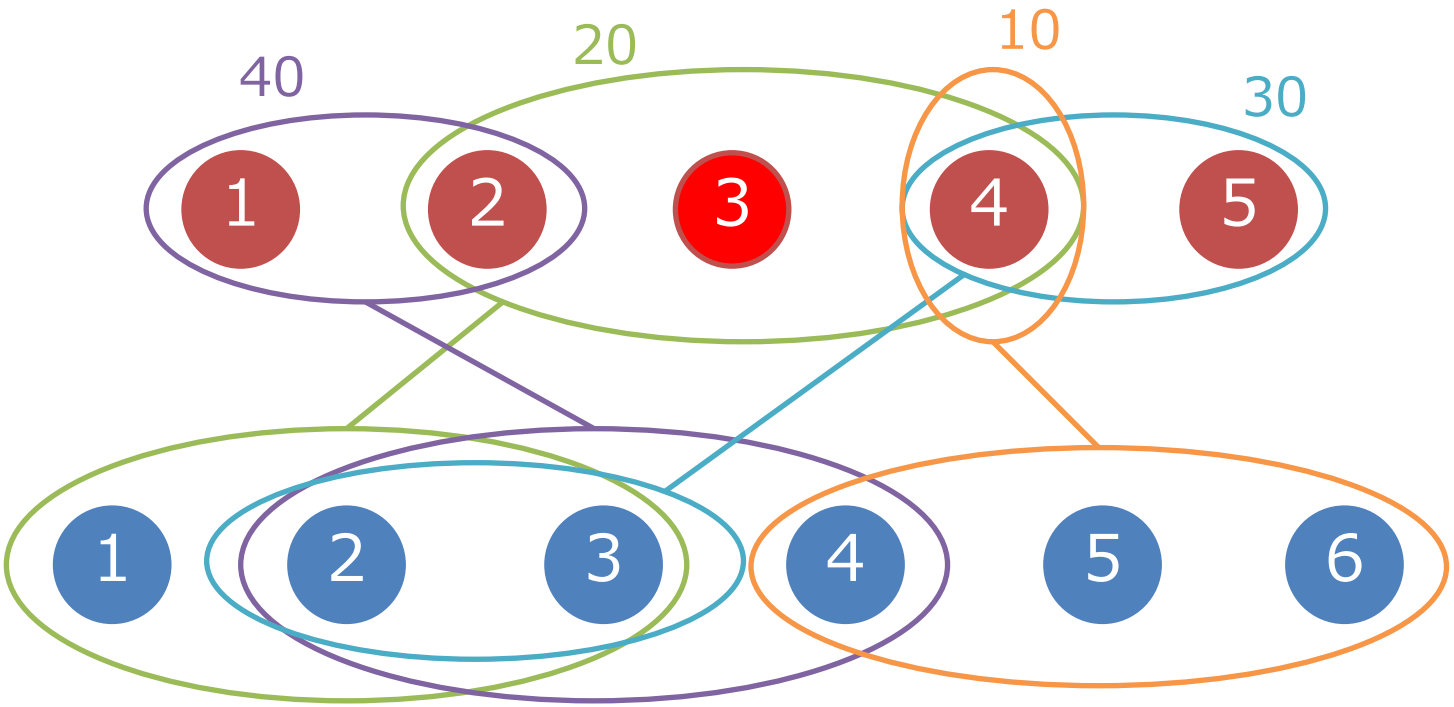
- 座標圧縮
  - 結構大変？
- $O(N^2)$
  - 50 点？

シミュレーション解法

- さらに列に対するデータ構造を使ってがんばる
  - Segment tree？
  - 超大変？
- $O(N \log N)$
  - 100 点？

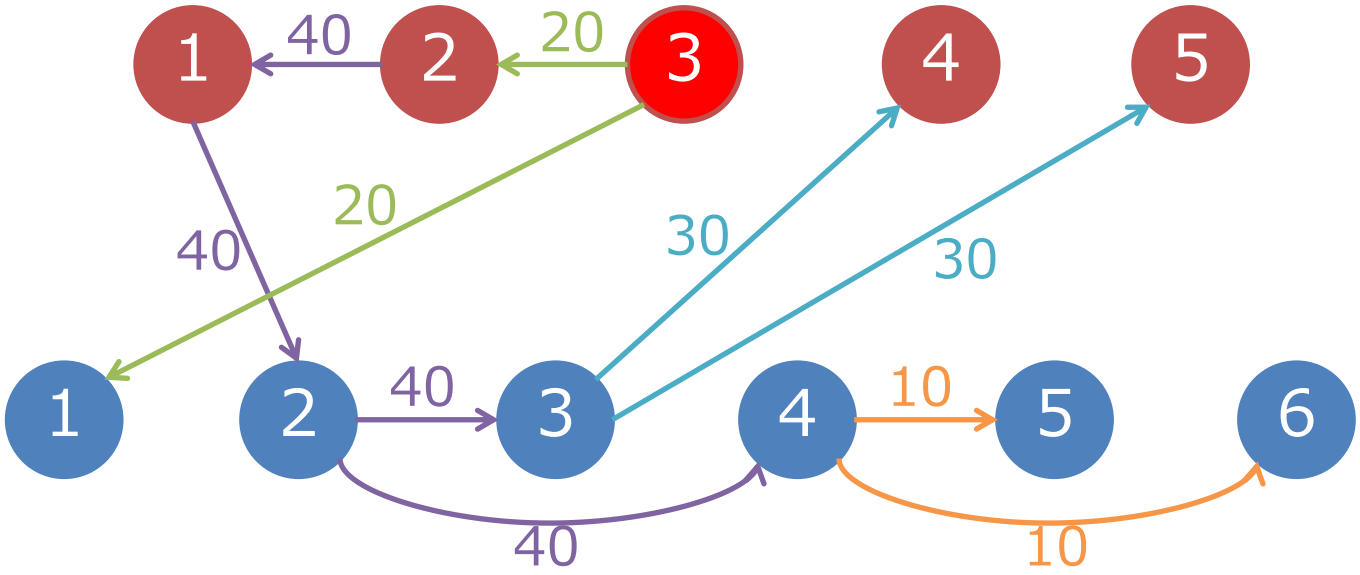
アイデア

- 誰のおかげで招待できたか？



アイデア

- 誰のおかげで招待できたか？



アイデア

- 誰のおかげで招待できたか？

- 全域木ができている
  - コストが対称なので無向グラフ

アイデア

- 問題文のアルゴリズム： 1 頂点から始めて，現在最大コストでとれる頂点を次々と繋げていく
- これは Prim 法そのまま

  →最大全域木が求まっている

  - C は関係ない
  - 番号の小さい方を優先，も関係ない

最大全域木

- 最大全域木のコストは Kruskal 法でも同じ答が求まる

- Kruskal 法
  - 辺をコストの大きい方からソート
  - 辺を順番に見ていき，異なる連結成分を繋いでいれば使う

最大全域木

- 仲良しグループを T_i の降順にソート
- グループには $O((\text{動物の数})^2)$ 本の辺があるが，それらを結ぶ $O(\text{動物の数})$ 本の辺だけを使えばよい
  - 犬 P_i と猫 R_i
  - 犬 P_i と犬 x ($P_i + 1 \leqq x \leqq Q_i$)
  - 猫 R_i と猫 y ($R_i + 1 \leqq y \leqq S_i$)

最大全域木

- 辺を繋ぐ処理は Union-Find を使う

- $O(N\ (A + B)\ \alpha(A + B))$
  - α ： Ackermann 関数の逆
    - 定数と思ってよい
  - 30 点

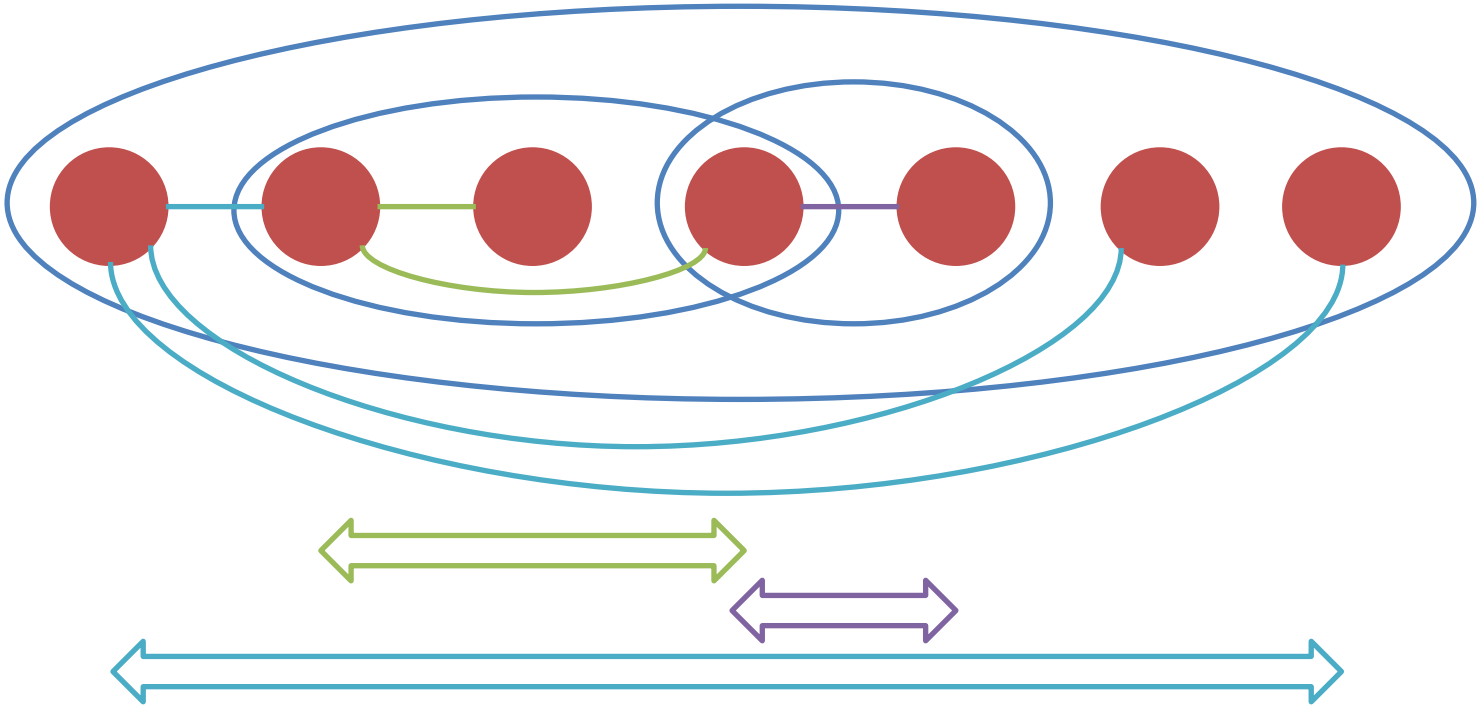
座標圧縮

- この方針なら座標圧縮は比較的容易

- 犬と猫を $O(N)$ 個の区間にわける
- 区間たちを Union-Find で繋いでいく
- 各区間は，最初に触れたときに内部を全部結ぶ
- $O(N^2)$
  - 50 点

高速化

- 各グループの辺を繋ぐのに $O(N)$ かかるのが無駄



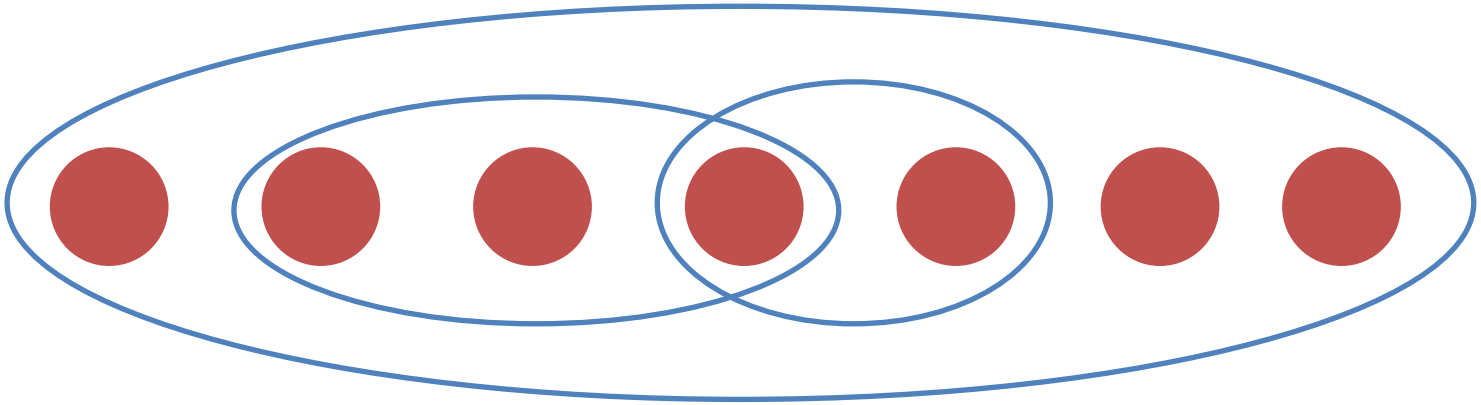
高速化

- 犬と猫の間の辺は 1 本でよいので実際に繋ぐ
- 犬どうし，猫どうしの辺で実際に繋ぐ必要がある辺は全部の合計は $O(N)$ 本

- 区間を繋げていくとき，すでに繋がっている部分はどんどんスキップしたい

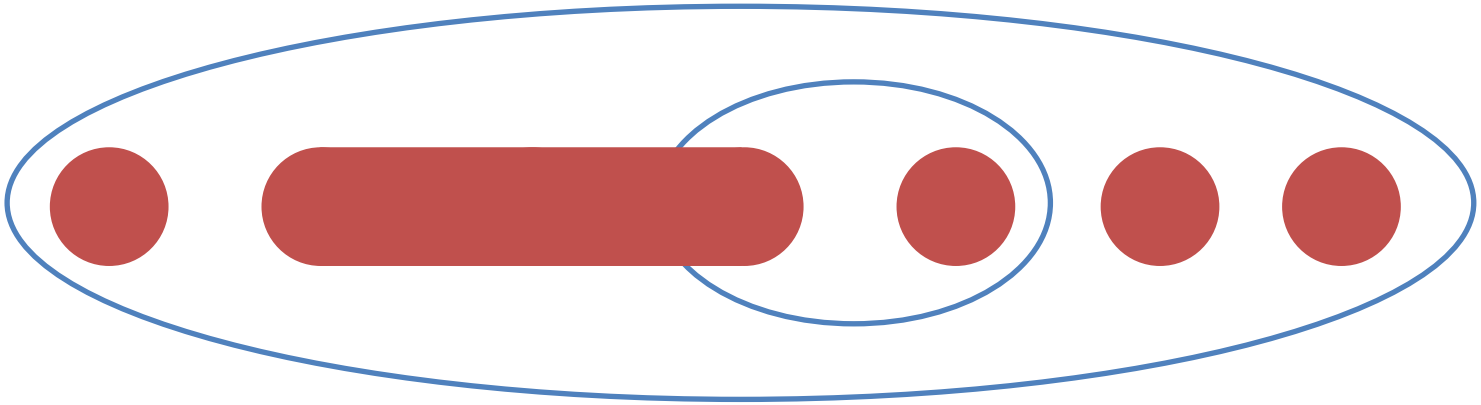
高速化

- 繋ぐたびに 1 つにしていく



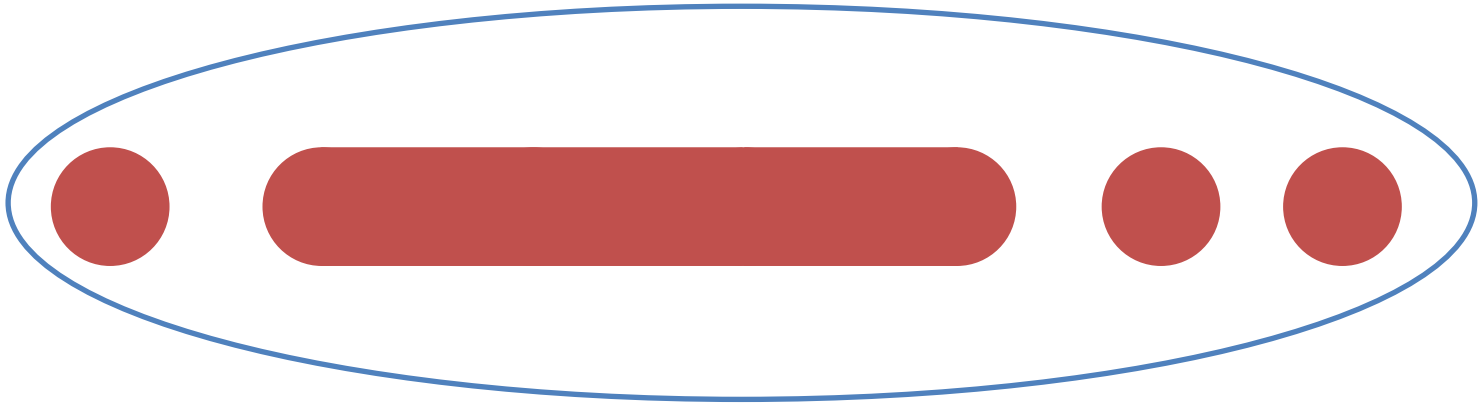
高速化

- 繋ぐたびに 1 つにしていく



高速化

- 繋ぐたびに 1 つにしていく



高速化

- 繋ぐたびに 1 つにしていく

高速化

- 繋ぐたびに 1 つにしていく

- 実現方法
  - 線形リストを使う
    - ちょっと複雑だが高速
  - std::set を使う
    - 実装は簡単
    - どうせ $O(N \log N)$ はかかるので

高速化

- 全体で $O(N \log N)$
  - 線形リストを使った場合は最初のソート以外は $O(N)$
  - 100 点

得点分布