D. Graph Construction

出題: 保坂
解答: 保坂
解説: 保坂

問題概要

- 無向グラフに対して 3 種類のクエリを処理する
  - (1) 頂点 u と頂点 v を結ぶ辺を追加
  - (2) 頂点 u と頂点 v を結ぶ辺を削除
  - (3) 頂点 u と頂点 v を結ぶパスがあるかどうかを判定して出力

- 頂点数 $n \leqq 40{,}000$
- クエリ数 $k \leqq 40{,}000$

ダメな解法

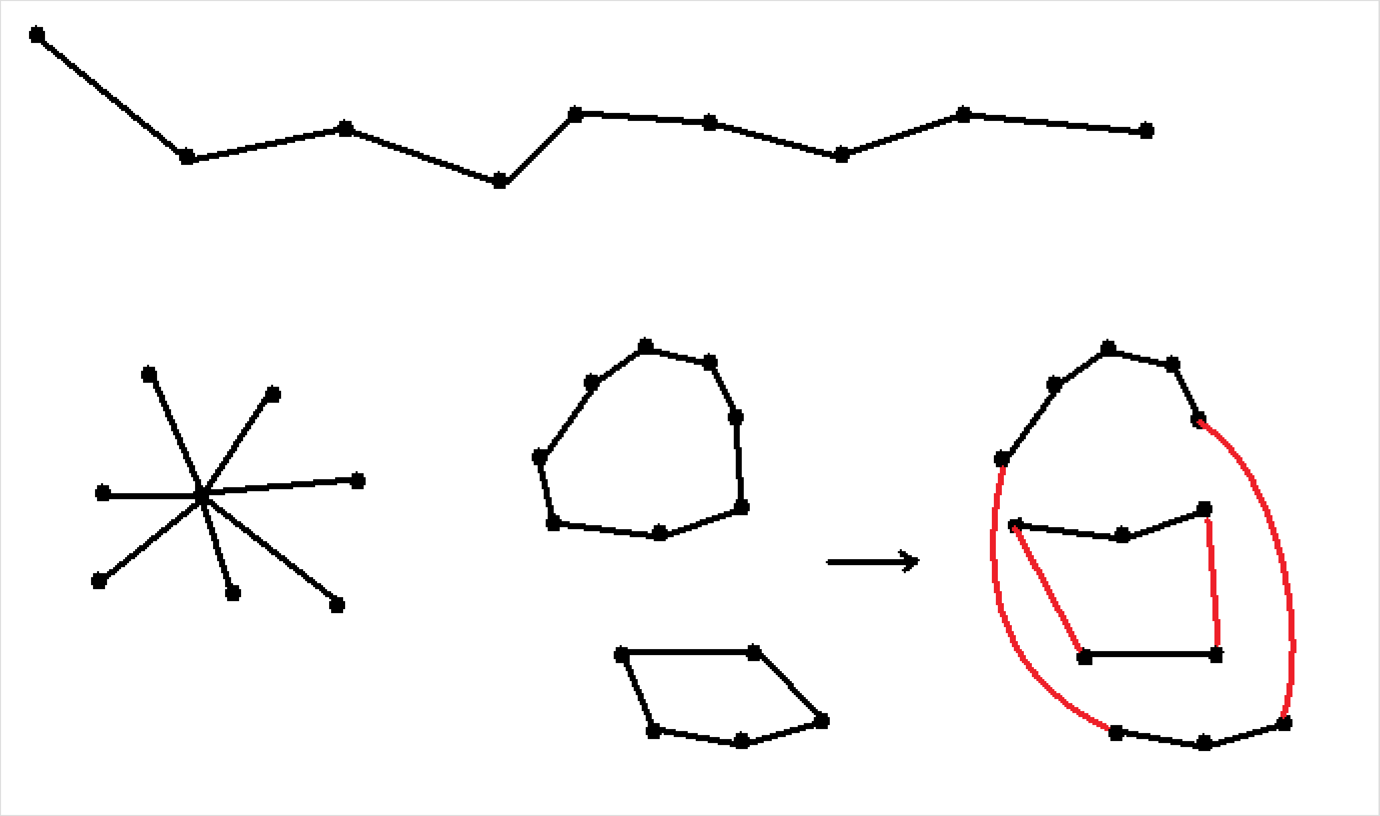
- 辺を適切に管理したうえで, クエリ (3) が来るたびに DFS などで行けるかどうか判定

ダメな解法

- 辺を適切に管理したうえで, クエリ (3) が来るたびに DFS などで行けるかどうか判定
  - $O(k^2)$
- 手軽に書けて, デバッグ用にも便利なので書いておいて損はないと思います

Judge Data

- O(k^2) の解法でも枝刈りなどで通る？
  - 怪しい＆ミスしやすい



クエリが 2 種類だったら…？

- 追加と判定だけ
  - (1) 頂点 u と頂点 v を結ぶ辺を追加
  - (3) 頂点 u と頂点 v を結ぶパスがあるかどうかを判定して出力

クエリが 2 種類だったら…？

- 追加と判定だけ
  - (1) 頂点 u と頂点 v を結ぶ辺を追加
  - (3) 頂点 u と頂点 v を結ぶパスがあるかどうかを判定して出力

- Union-Find を使えばよい
  - $O(k\,\alpha(k))$

クエリが 2 種類だったら…？

- 削除と判定だけ
  - (2) 頂点 u と頂点 v を結ぶ辺を削除
  - (3) 頂点 u と頂点 v を結ぶパスがあるかどうかを判定して出力
- 最初に辺がいくつか与えられる

クエリが 2 種類だったら…？

- 削除と判定だけ
  - (2) 頂点 u と頂点 v を結ぶ辺を削除
  - (3) 頂点 u と頂点 v を結ぶパスがあるかどうかを判定して出力
- 最初に辺がいくつか与えられる

- 「うさぎたちは, 昔似たような問題を解いたので」
  - 夏合宿 2009 Graph Destruction

解法？

- Union-Find をちょっと変えればよい？

解法？

- Union-Find をちょっと変えればよい？
  - Union-Find の特徴を考えると, 計算量を log 程度までに抑えたまま辺を削除するのは無理そう

解法？

- Union-Find をちょっと変えればよい？
  - Union-Find の特徴を考えると, 計算量を log 程度までに抑えたまま辺を削除するのは無理そう

- 何かもっとすごいデータ構造がある？
  - 動的木…？

解法？

- Union-Find をちょっと変えればよい？
  - Union-Find の特徴を考えると, 計算量を log 程度までに抑えたまま辺を削除するのは無理そう
- 何かもっとすごいデータ構造がある？
  - 動的木…？
  - 知りません
  - きっと実装が大変で短時間のコンテスト向きではない

  - 一応何かあるらしいです

想定解法

- 最初にすべてのクエリを読み込んでしまうことが可能
  - Union-Find の利点だったオンライン性を諦める

想定解法

- 最初にすべてのクエリを読み込んでしまうことが可能
  - Union-Find の利点だったオンライン性を諦める

- メインアイデア: **平方分割** (平方根分割？)
  - クエリを √k 個ずつの √k 個の区間にわけてみる

想定解法

- O(k √k α(k)) くらいなら大丈夫
  - 以下 α(k) は 1 とみなす

想定解法

- O(k √k α(k)) くらいなら大丈夫
  - 以下 α(k) は 1 とみなす
- 区間は √k 個
  - 各区間に対して O(k) で処理すればよい
- クエリ (3) は O(k) 個
  - 各クエリ (3) に対して O(√k) で答えればよい

区間に対する処理

- 追加される各辺に対して, 追加される時刻と削除される時刻を記録
  - 削除された後に復活したら別の辺とみなす

区間に対する処理

- 追加される各辺に対して, 追加される時刻と削除される時刻を記録
  - 削除された後に復活したら別の辺とみなす
- 各区間について,「常に存在する辺」を処理する
  - Union-Find など
  - O(k)
  - このときの各連結成分を 1 つの頂点だと思ったグラフを考える

区間に対する処理

- 「常に存在する辺」による連結成分たち
  - 区間中に, クエリ (1), (2), (3) として全く現れない頂点しか含まないような連結成分は消すことができる
  - 残る連結成分は高々 $2\sqrt{k}$ 個
  - 区間中に現れる頂点は番号を付け替える

クエリ (3) に対する処理

- 連結成分をまとめたグラフについてさらに Union-Find などで処理
- 区間中のクエリ (1), (2) として登場する辺のうち, 追加時刻と削除時刻が答えたいクエリ (3) を挟んでいるようなものを追加すればよい
  - 高々 $\sqrt{k}$ 本の辺

高速化？

- 半分のサイズのグラフに対して再帰的に解く
  - 真面目に実装すれば, サイズ m の区間に対して O(m) 時間なので, 全体で O(k log k)
  - 適当にやると $O(k (\log k)^2)$ とかになりがち
  - 定数が重いです

結果

- 正解 / 提出:　0 / 2
- 提出チーム:　1 / 8
- 正解チーム:　0 / 8
- 最初の提出:　_ (04:01)