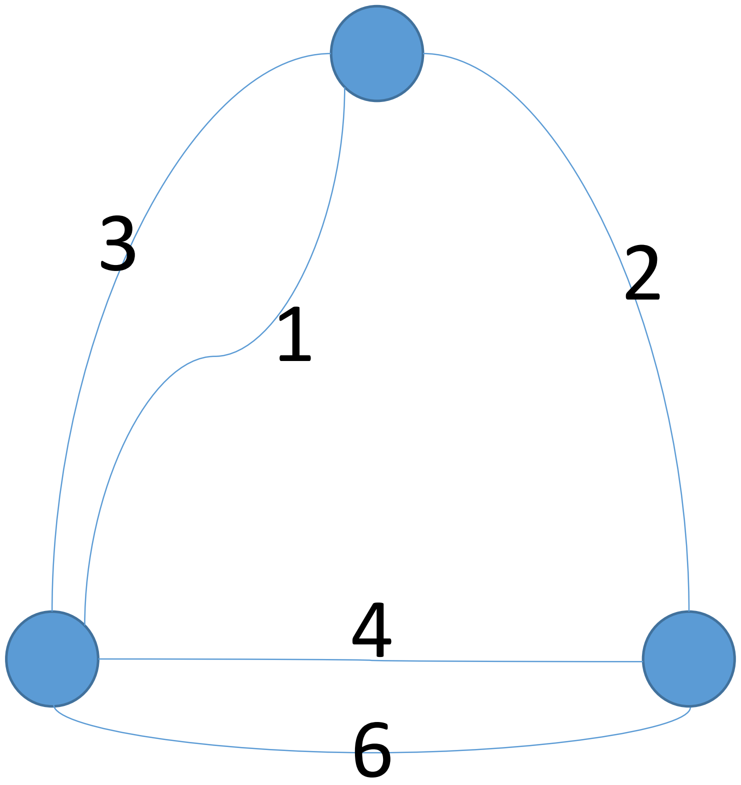
遺産相続（Inheritance）

笠浦一海

問題概要

- 頂点数N(≦1000)、辺数M(≦300000)のグラフがある。
- 辺には重みがついてる
- 『閉路を含まない』という条件下で重み最大の辺集合を見つけ、それをグラフから取り除く
- 个をK(≦10000)回おこなう
- それぞれの辺が何回目に取り除かれたかをこたえよ

問題概要（例）



子供1：3 + 6 = 9

子供2：4 + 2 = 6

子供3：1

条件を言い換える

- 辺の重みはすべて正　→　取れる辺がなくなるまで辺をとる
- 『閉路を含まない』という条件でたくさん辺をとりたい
  →　各連結成分に対して全域木をとればよい
- 最大全域木をもとめる
  →　辺の重みを × (-1) して最小全域木を求める
  →　Kruscal 法，Prim 法

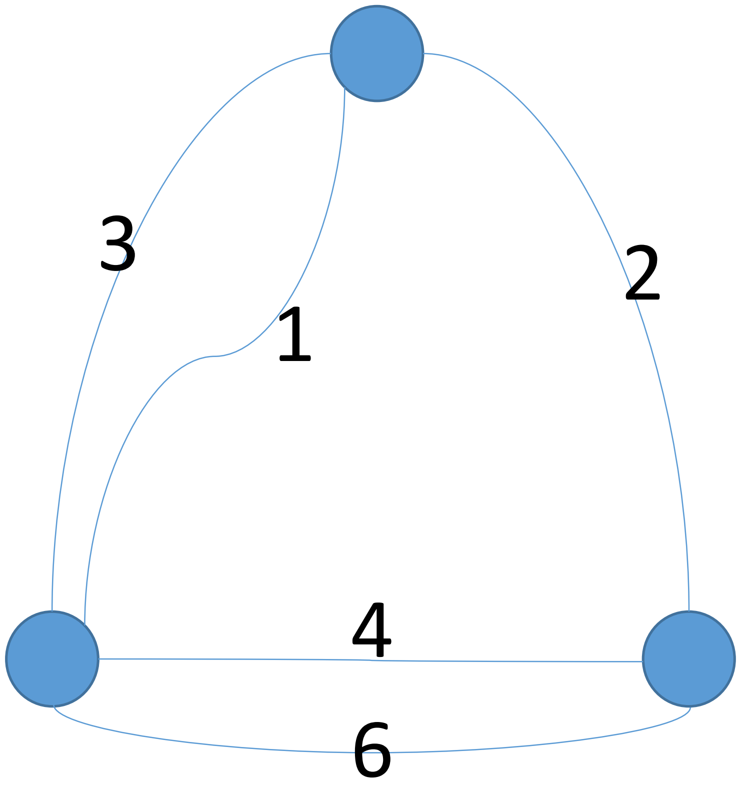
プリム法

- 詳細は蟻本などを参照
- 隣接行列で実装：O(E + V^2)
- vectorとpriority_queueで実装：O(E log V)
- この問題では K 回行うので O(KM + KN^2) か O(KM log N)
- 前者はうまく実装すると O(M log M + KN^2)
- 隣接行列のほうが早い
- なんにしろ15点

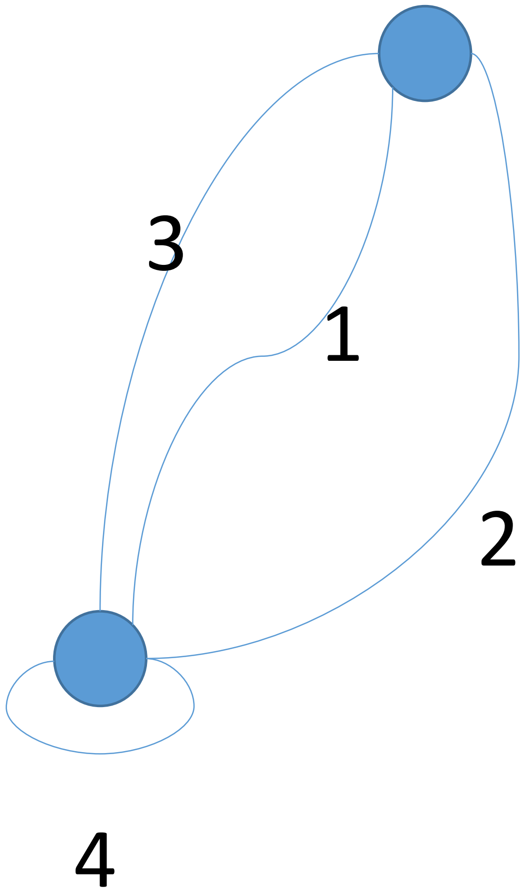
クラスカル法

- 重みの大きい辺から貪欲にとっていく
- ただし閉路ができてしまう場合は飛ばす
- 閉路ができるかの判定は Union Find を使う

クラスカル法

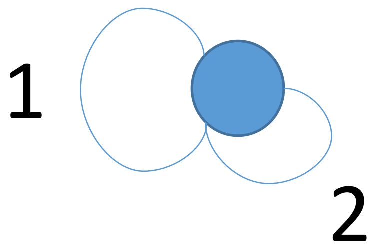


クラスカル法



子供1:6

クラスカル法



子供1：6+3

クラスカル法

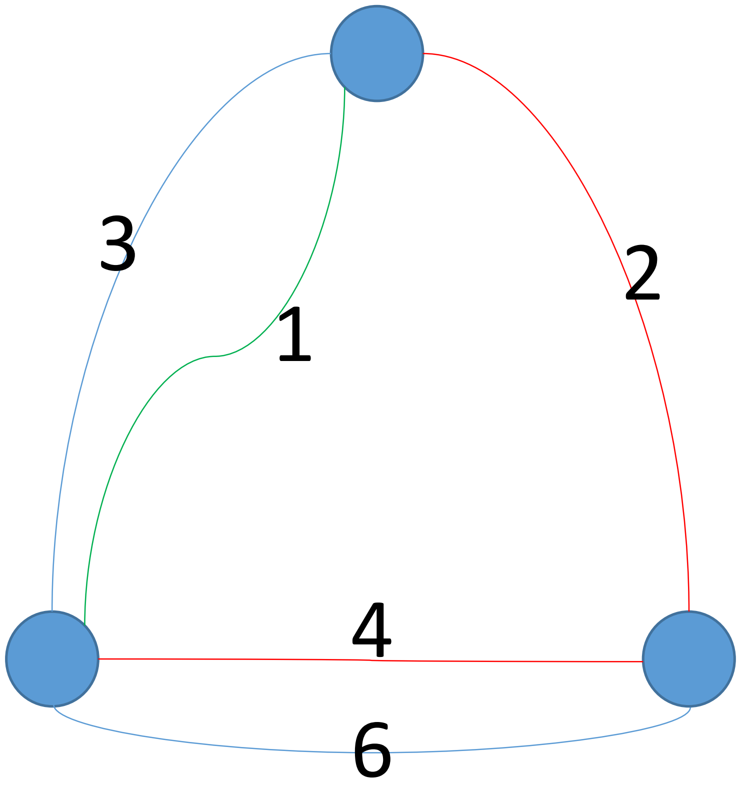
- 計算量は辺のソートに O(E log E) 貪欲にとるのに O(E α(V))
- ただしαは逆アッカーマン関数という増加の遅い関数
- 辺のソートは最初に1回行えばいいので、この問題では計算量は O(M log M + KMα(N)))
- Subtask 1 のみ15点

クラスカル法解法を改変

- 各子供ごとに、辺を渡るループを回す
- 順番を変えて、各辺に対して、子供を渡るループを回してもよい
- すなわち重みの大きい辺から順番に、誰に相続されるかを決めていく
- 子供1から順番に見て行って、最初に相続できる人を見つける
- 相続できる⇔追加しても閉路ができない

⇔その辺が結ぶ二頂点はつながっていない

クラスカル法解法を改変(例)



辺6：子供1○

辺4：子供1×→子供2○

辺3：子供1○

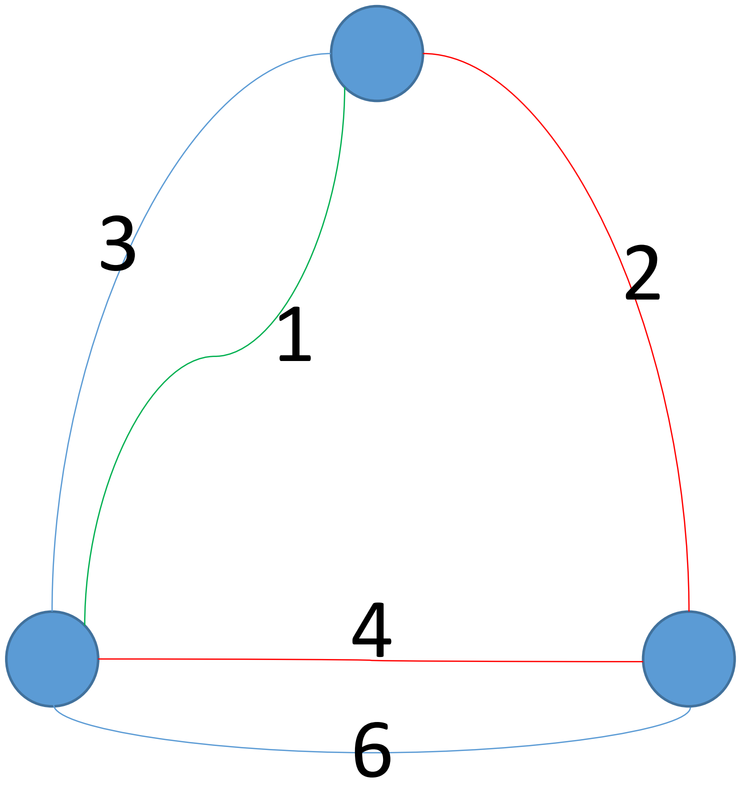
辺2：子供1×→子供2○

辺1：子供1×→子供2×→子供3○

クラスカル法解法を改良

- K個のUF木が必要→空間計算量O(KN)が必要
- このままだとまだ計算量は$O(M \log M + KM\alpha(N))$
- 相続する人を**二分探索**で探せばよさそう

二分探索(例)



辺6：子供2○→子供1○

辺4：子供2○→子供1×

辺3：子供2○→子供1○

辺2：子供2○→子供1×

辺1：子供2×→子供3○

二分探索の正当性

- 二分探索は正当か？
- 「子供 i が相続できる　⇒　子供 j も相続できる ($i < j$) 」が成り立てばよい
- すでに相続先が決まっている辺のうち、子供 i が相続する辺の集合をT_iとおく
- 「aとbがT_jによってつながっている　⇒　aとbがT_iによってつながっている ($i < j$)」　が任意の時点で成り立てばよい

二分探索の正当性の証明

- これが成り立つことを帰納法で示す
- 初期状態では明らかに成り立っている
- 頂点c，dを結ぶ辺αがT_kに追加されるとする
- このときkは相続できる子供の番号の最小値なので、c，dは$T_1,\ldots,T_{k-1}$によってつながっているはずである
- よって「aとbがT_jによってつながっている ⇒ aとbがT_iによってつながっている ($i < j$)」の条件は保たれている
- Q．E．D.

満点解法

- どの子供が相続するか各辺に対して二分探索を行う。
- 時間計算量 : $O(M \log M + KN + M \log K\ \alpha(N))$
- 空間計算量 : $O(M + KN)$
- 時間はOK
- K<=10^5, N<=10^3 より KN<=10^7
- UF木はintの配列だけで実装できる
- 大きさ N の int の配列 K 個で十分なので空間計算量もOK

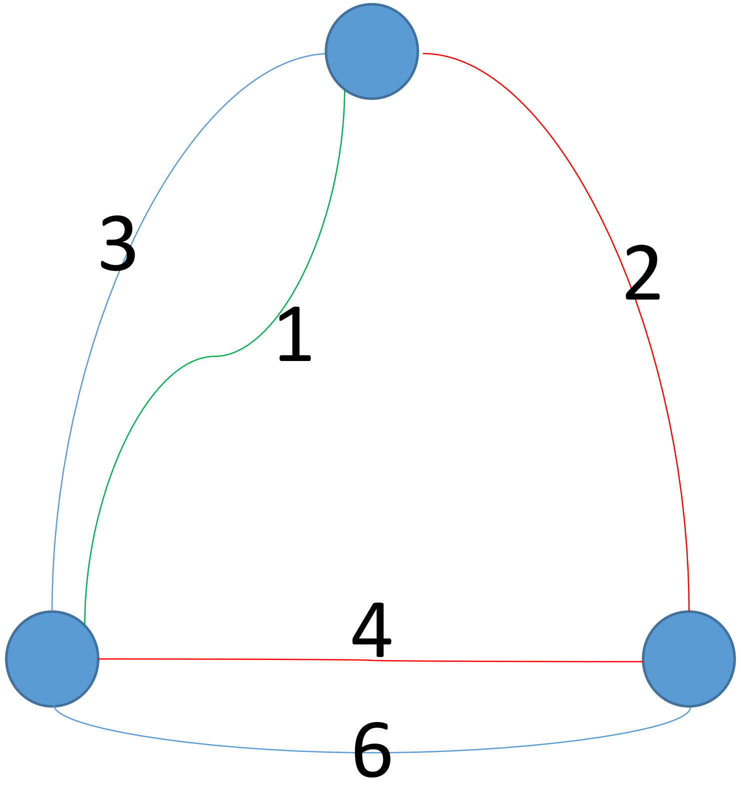
＋α

- UF木のために KN の領域を確保するが、そのうち実際の計算で使うのは O(M log K) のみ
- 動的にメモリを確保すれば節約できそう
- はじめに大きめの配列を確保し、配列の番号をmapで管理
- 各子供ごとにmapを用意すれば、それぞれのmapの大きさは高々N
- 時間計算量：O(M log M + M log K log N)
- 空間計算量：O(K + M log K)
- 実装をうまくやらないと間に合わない

別解

- 基本方針は満点解法と同じ
- 二分探索を行う代わりに次の高速化を行う
- 頂点a,bを結ぶ辺αが子供 i に相続されたとする
- 次に頂点a,bを結ぶ辺βが出てきたとき、子供1,…,i は確実に相続できない！
- 各頂点のペア{a,b}に対し、a,bを結ぶ辺のうち、直前に出てきたものを相続した子供の番号を記録
- 次にa,bを結ぶ辺が出てきたときは、その次の番号から調べる

別解(例)



辺6：子供1〇

辺4：子供2〇

辺3：子供1〇

辺2：子供1×→子供2〇

辺1：子供2×→子供3〇

別解

- 実はこれだけで計算量がへる！
- O(KN + N^2 + M√Mα(N)) とかになる
- (評価の仕方はいろいろある)
- ポイント:「頂点a,bを結ぶある辺を子供 i が相続するか調べる」事象を考える
- a,b,i について 1 回きり　→　O(K N^2) はわかる

別解

- この事象が起きるとき、最終的に子供 i が相続する辺において a,b はつながっている
- よって i を固定した時の a,b の場合の数は(子供 i が相続する辺の数+1)^2 で抑えられる
- これを利用してうまく評価すればよい（詳細は略）

別解(いま考えた)

- O(KN^2) のプリム法を高速化
- 各頂点に対し、連結している頂点の集合をsetかvectorで管理
- 同じ理由で早くなるはず……？

得点分布