F:marukaite

原案:岸本(tokoharu)
問題文:岸本(tokoharu)
解答:岸本(tokoharu),汐田(shioshiota)

問題概要

- n*nの大きさの格子状のマス目(方眼?)に丸が書いてあったら書いて無かったりする
- 各行各列に一つだけ丸印があるようにしたい
- 各行各列に丸を書くコスト、丸を消すコストがそれぞれ与えられる
- 目標までのコストを最小化する。加えて、そのコストを達成するような手順も出力。
- N<=100 ， 1<=cost<=1000

考察1

- 無駄なところ
  - コストはすべて1以上なので、同じ地点を書いて消すor消して書く、という操作は無駄。
  - だから丸印があるところは消去コスト、丸印が無いところは書き込みコストしか用いない。
- 全探索は?
  - n=100のときにどうやっても死亡

考察2(と後のページのヒント)

- 結論から言うと、行と列を対応させる最小費用流です
  - なので張る辺のコストを考えましょう。
- 最小費用流とは
  - 調べてください(ごめんなさい)
  - どういう問題か。どう解けるか、などをチェックしてみてください
- 後のページへのヒントだけ
  - 後のページでbellman-ford , dijkstraの最短経路問題を解くためのアルゴリズムの名称が登場します
  - これは、最小費用流を求めるために、内部で最短経路問題を解く必要があるからで、自分のアルゴリズムによってどちらを使うべきかも考える必要があります。

考察3 (準想定解法p. 1)

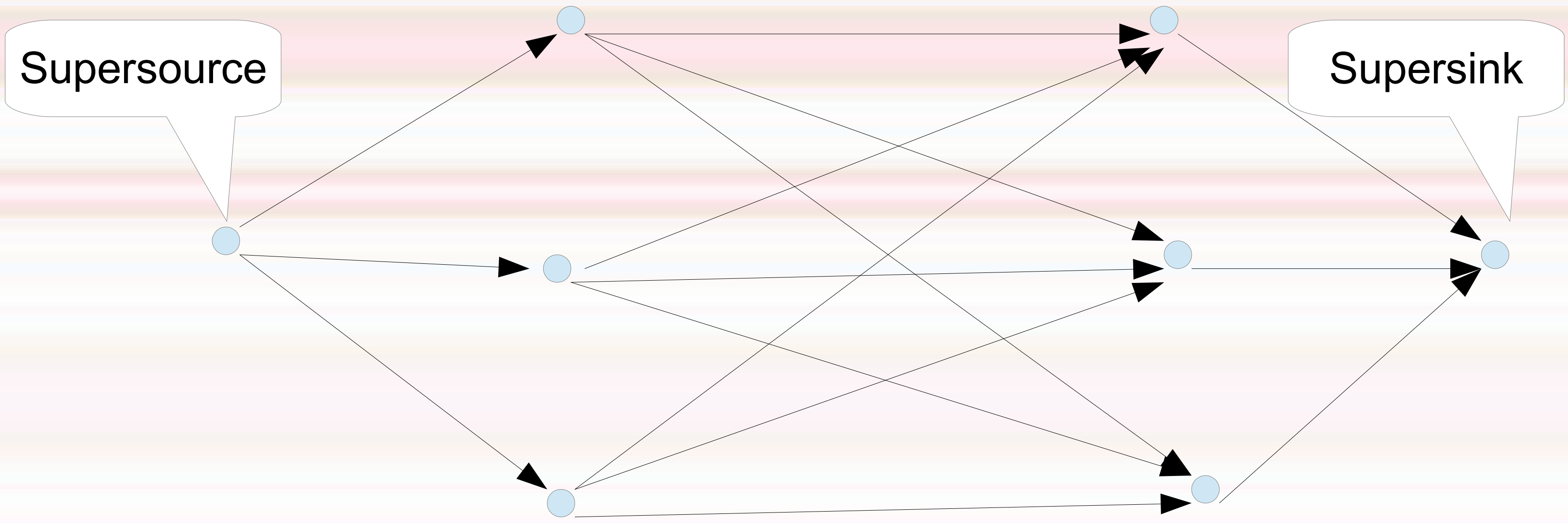
- まず、丸印をなかったことにします。なので、いったん全部の消去コストを足し合わせます(和をCとします)。

- このとき、ある行ある列に丸印があるようにするには、
  - 1.もともと空白なら、書き込みコストを足し合わせる
  - 2.もともと丸印があるなら、先ほど使った消去コストを無かったことにするので消去コストだけ引く
  - だけのコストが必要、と考えれます

考察4 (準想定解法p. 2)

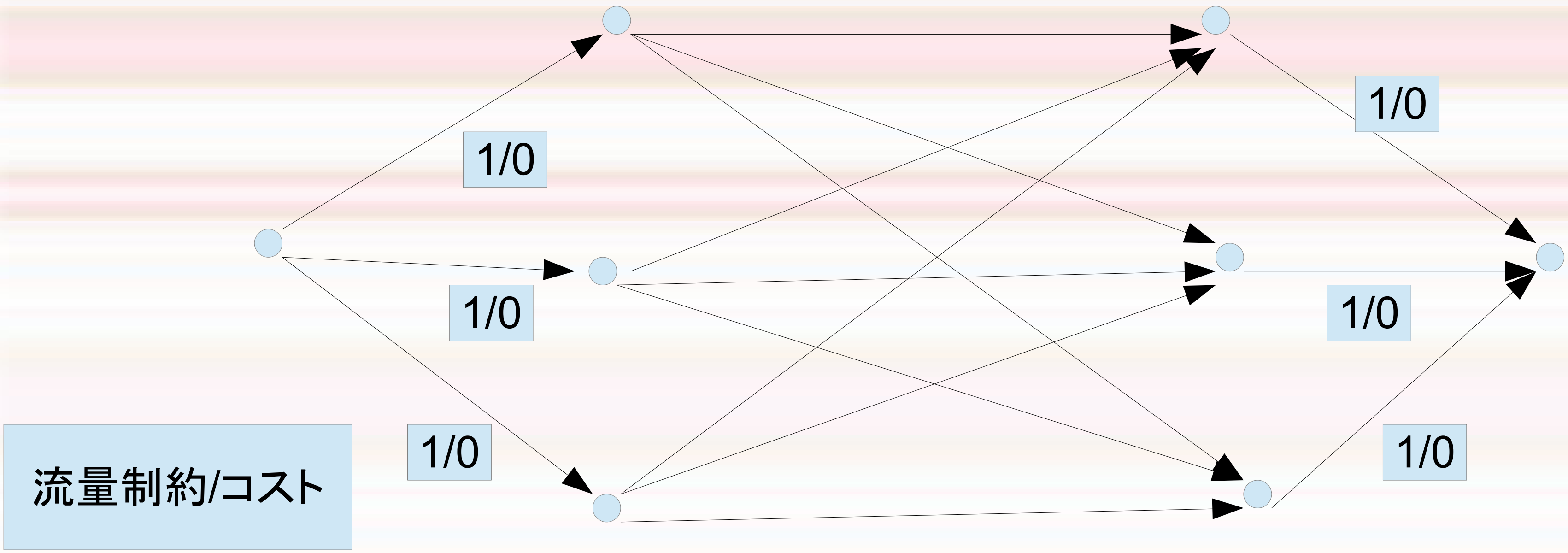
- 前のページのコストを用いてグラフを構築します
  - 言葉だけではわからないので後ページの図を参照
- どんな最小費用流にするか
  - 負のコストがあるので最小費用流内部の最短経路問題は bellman-fordの方を用いる
- 解析
  - V=200,E=100^2くらいのグラフを構築できる
  - O(FVE) = O(N^4)くらい。
  - 代入すると2億そこら。AOJでは間に合いそう
  - CodeChefなら死にそう(偏見)
- 毎回bellmanで最長だけかけるテストケースを作る技術を持ってなかったこともあり、余裕を持って通ります

考察2のときに作られるグラフ *p. 1*



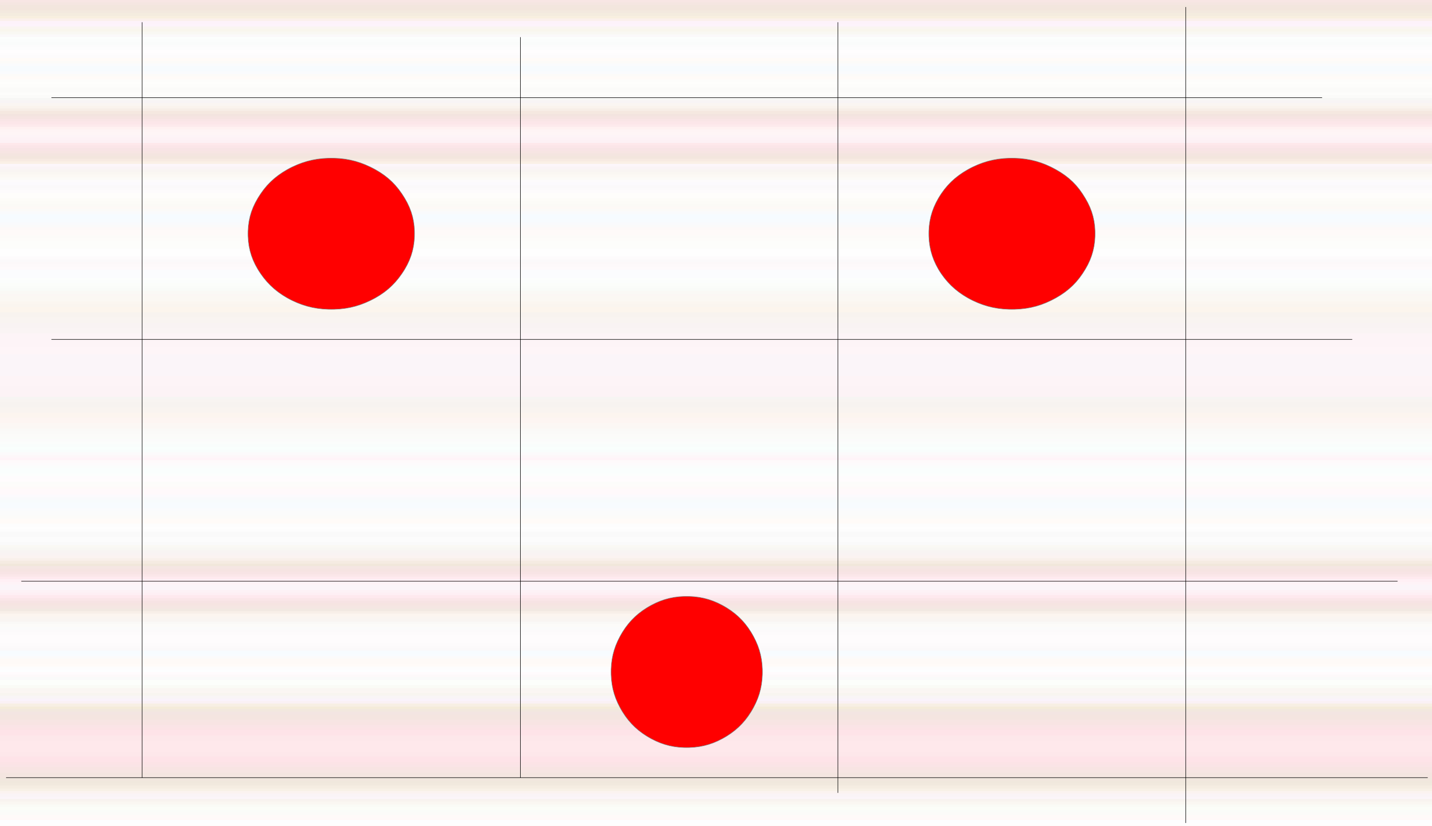
- 作られるグラフの概形は、以上のような二部グラフにに左右の点を加えたようなものです。
- 左端、右端の点はそれぞれ、supersource ， supersinkと呼ばれる水を流し始める場所、水を流し終える場所を意味しています

考察2のときに作られるグラフ *p. 2*



- supersourceから行を表す頂点, 列を表す頂点からsupersinkへの辺は、流量制約1, コストは0とする。
- これをすることにより、最終的にnの流量を流し、それぞれの辺に必ず1ずつ流す状況にすることで「各行各列1だけ丸印がある」条件を的確に表現できる。

資料*1*



資料2

-1	1	-1
1	1	1
1	-1	1

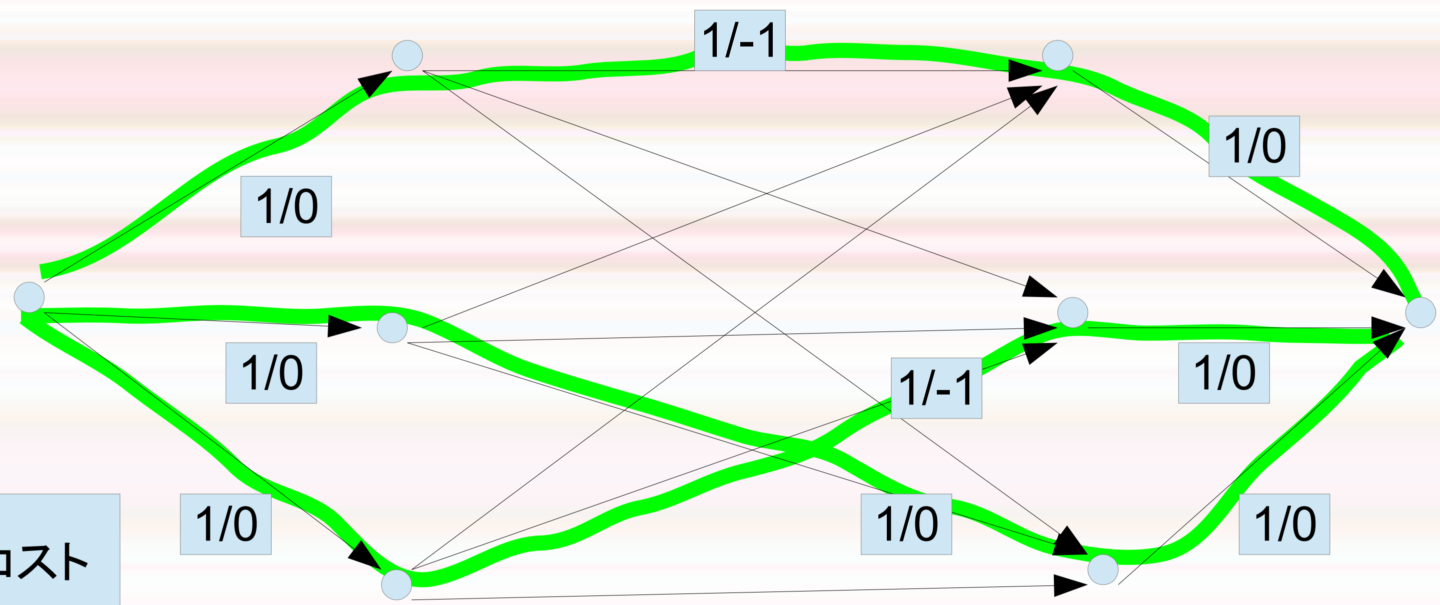
コスト : 3

資料3

-1	1	-1
1	1	1
1	-1	1

コスト：3 + (-1) + (-1) + 1 = 2

考察2のときに作られるグラフ *p. 3*



流量制約/コスト

- 前頁に出てきた例では実際には以上のように流れることになる。

考察5 (想定解法1p. 1)

- 本当に負コスト必要ですか？
  - 各行で丸印は1回しか使わない
  - かかるコストを行ごとに分割して考えてみる
  - ある行iで、丸印をなかったことにする。この時かかるコストをCiとする。
  - 行iに対してある列jに丸印が存在するとする。このとき、
    - 1.もともと丸印がある → Ci-erase[i][j]
    - 2.丸印が無い → Ci+write[i][j]
    - を辺のコストにする。

考察6(想定解法1p. 2)

- 前頁のようにすれば全部正のコストになります
- Dijkstraでも大丈夫！
- 計算量
  - O(FElogV) = O(N^3logN)

考察7(想定解法2/ハンガリアン法)

- 良く考えると行と列に対応するコスト決まったら割り当て問題と考えれる
- じゃぁハンガリアン法をつかおう！(中国のほうではKM算法と呼ばれてるらしい)
- ナイーブにやるとO(N^4)だそうですが、ちゃんとやればO(N^3)になります(想定解法2)
- 走らせてみると一番高速でした。

まとめ

- 解けなかった人
  - フローに帰着できる場合、コストをうまいこと操ってうまく容量条件を付けるとうまくいきます
  - 解けなかった人はいろんな問題に触れましょう。
  - 今回のように列と行を対応させて頑張るのは良くあるパターンです。要復習です
- ハンガリアン法知らなかった人
  - ぜひこの機会に知っておきましょう。
  - 割り当て問題は費用流でも解けますが、負コストでもハンガリアンで解けたとおもいます
    - 怪しいのでJAGの人教えてください
  - 余裕があればライブラリに。

フローにはどこで巡り合えますか?

- アルゴリズムデザインでは最大流の項目だけで一章まるまる使ってます(DPと同列!!最小費用流にすら触れてないのに!!)
- DPに比べると問題をそこまで多くは見かけませんが、探すと見つかると思います(SPOJにはtoolのところに行けばclassificationというところでちょっと分類されてます)
- 筆者(tokoharu)はどこに行けばフロー系の問題と戯れることができるのかあまりよく分かっていないので、オススメがあれば是非教えてください。

おすすめフロー問題

- ZOJ3569 Dr. Zomboss's Revenge
  - 辺の張り方が好きです
- CodeChefLong Sep2012 PARADE
  - 難しいですが、フローで良く見るようなテクニックを多用しています。

Judge解概要

- 汐田(準想定/mincost+bellman):168行
- 岸本(想定1/mincost+dijkstra):186行
- 岸本(想定2/ハンガリアン):161行

提出状況

- (注意：この問題はコンテスト本番では途中からAvailableな問題になりました)

- AC/SB: 11/19
- FA: wakaba (---min)