Warshall-Floyed法

Warshall-Floyed法とは

- ワーシャルフロイド法
- 全点対間最短経路(長)を求めるアルゴリズム
- 全点対最短経路: 任意の2点間の最短経路
- 計算量は$O(|V|^3)$
- 実装がすごい簡単

基本的な考え方

ある経由地点を固定してそこから任意の2点への最短経路を更新していく

dist[i][j] := (iからjへの最短経路長)

1. 初期値

$$dist[i][j] = \begin{cases} 0 & i = j \\ \infty & i \neq j \end{cases}$$

2. kを固定
3. 任意の点i,jについて,
   kを経由した経路dist[i][k] + dist[k][j]
   が最短経路になるか確認しdist[i][j]を更新

実装

- 以下頂点数nとする.

```
// 初期化
for (int i = 0; i < n; i++) {
  for (int j = 0; j < n; j++) {
    if (i == j) dist[i][j] = 0;
    else dist[i][j] = INF;
  }
}
```

- **3重ループ書くだけ.超簡単.**
- k,i,jの順に注意

```
// 辺の追加
// iからjへコストcの辺を張るときは
// dist[i][j] = c と書く.

// Warshall Floyed法
for (int k = 0; k < n; k++) {
  for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
      if (dist[i][k] == INF || dist[k][i] == INF) continue;
      dist[i][j] = min(dist[i][j], dist[i][k] + dist[k][j]);
    }
  }
}
```

実際の動き

ここに載せるには画像が多すぎる.

- 全パターン画像作ると膨大になるため，一部省いています
    - imgフォルダに入れておきます
- グラフの●: iとj
- グラフの●: ik
- 表の■: 辺i->k
- 表の■: 辺k->j
- 表の■: 辺i->j

負閉路検出

Warshall-Floyed法を行った後，
$dist[i][i] < 0$ となることがあれば，負閉路を持つ.

応用

- 制約が間に合うならWarshall-Floyedが強い