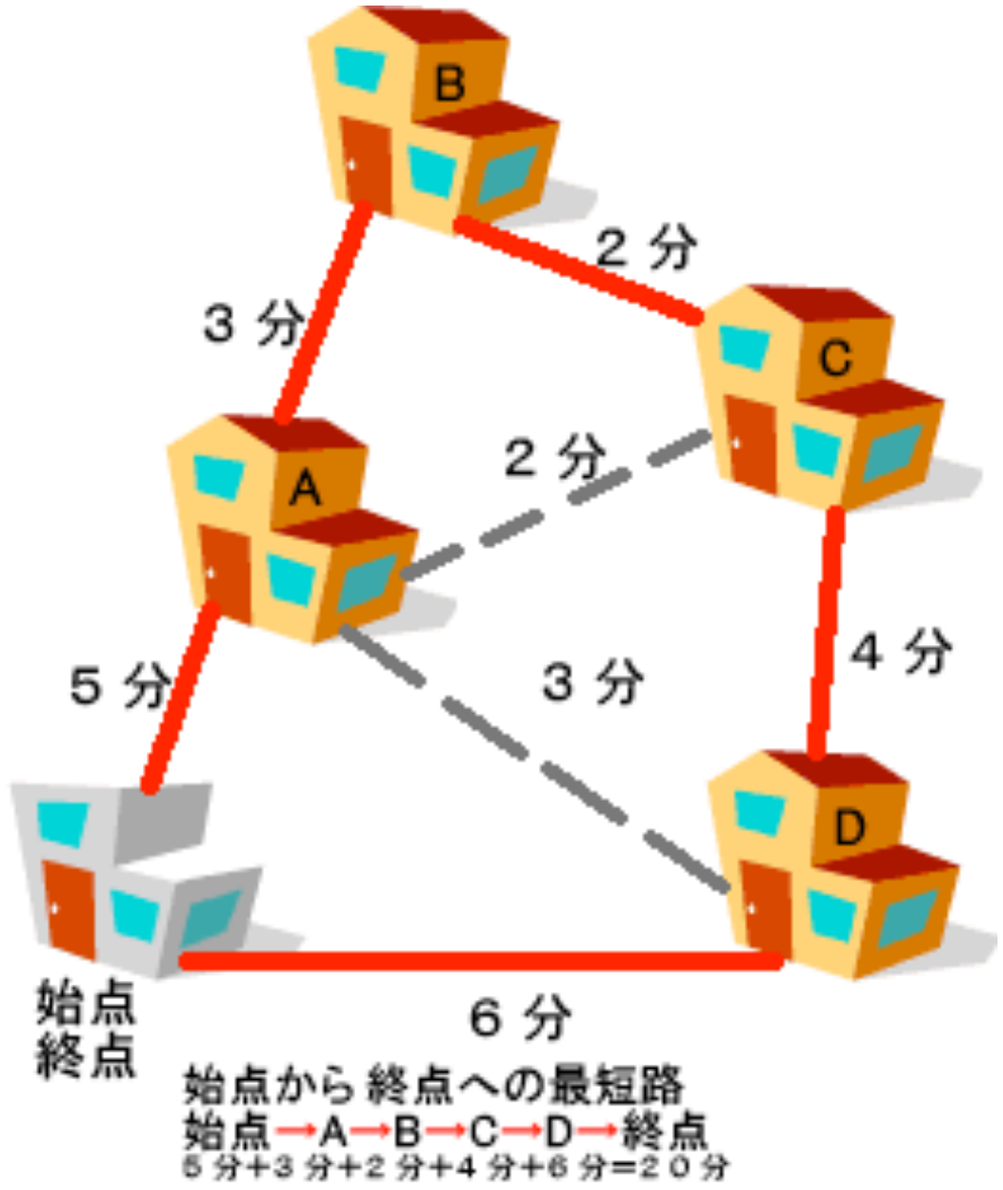
競プロ勉強会03

elzup

最短経路問題

- 最短経路となるルートを求める

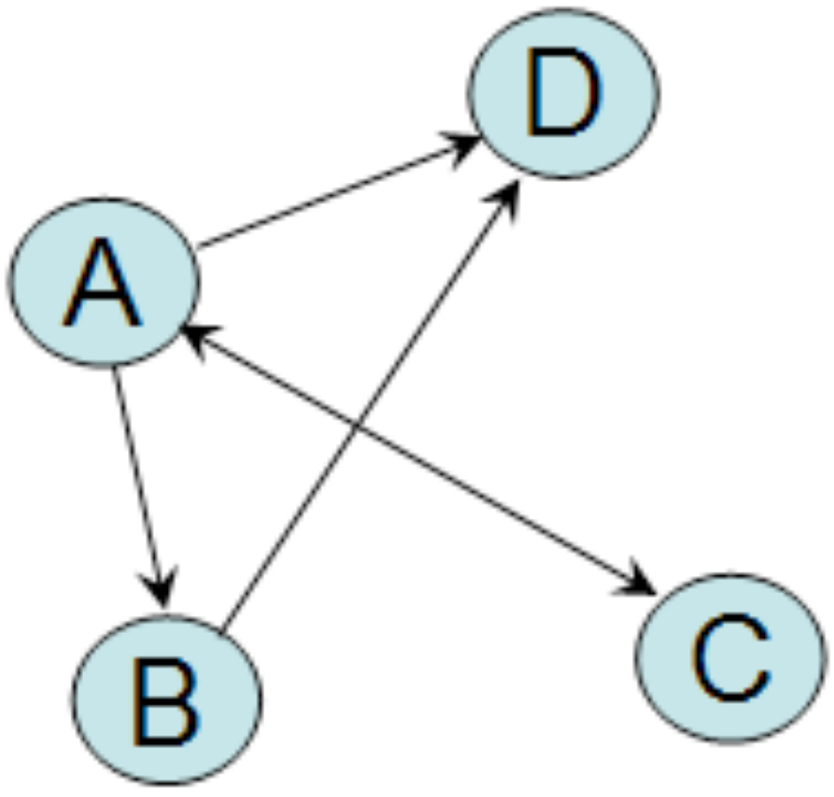
例: 巡回セールスマン問題



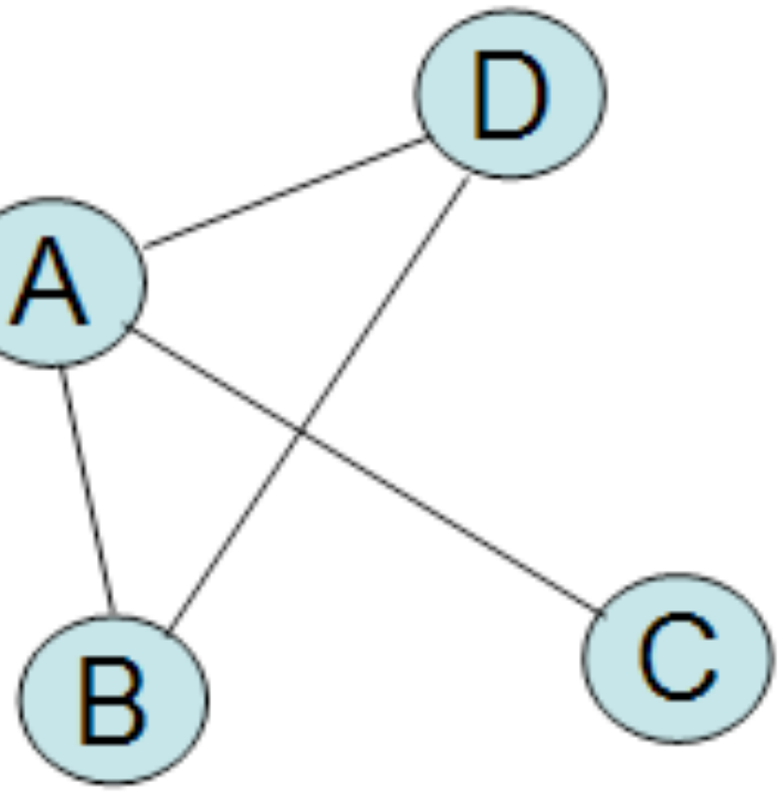
2つ

- ダイクストラ
  - 2頂点対最短経路問題
  - 単一始点最短経路問題
- ワーシャルフロイド
  - 全点対最短経路問題

グラフについて



(a) 有向グラフ



(b) 無向グラフ

ダイクストラ

- 単一始点最短経路問題
- n log(n)
- 図解: http://www.deqnotes.net/acmicpc/dijkstra/

解いてみよう

- 問題: Single Source Shortest Path
- http://judge.u-aizu.ac.jp/onlinejudge/description.jsp?id=GRL_1_A&lang=jp

使うデータ構造

```
    static class Edge {
        public int source;
        public int target;
        public int cost;

        public Edge(int source, int target, int cost) {
            this.source = source;
            this.target = target;
            this.cost = cost;
        }


    }
```

入力受け取り

- ＶＥＲ
- s0 t0 d0
- s1 t1 d1
- ：
- s(E-1) t(E-1) d(E-1)

その他アルゴリズム

- 最短経路
  - ベルマンフォード
- 探索
  - A*
  - 幅優先探索

Edge class

```
static class Edge {
    public int source;
    public int target;
    public int cost;
}
```

- int[] distance = new int[vn];
- Arrays.fill(distance, INF);
- distance[s] = 0;
- Queue<Edge> queue = new PriorityQueue<>();
- queue.add(new Edge(s, s, 0));

```
·       Edge e1 = queue.poll();                    // 今回チェックするノード
·       if (distance[e1.target] < e1.cost) {  // 更に小さい値で更新済みはスキップ
·           continue;
·       }
```

```
·       for (Edge e2 : edges.get(e1.target)) {
·           if (distance[e2.target] > distance[e1.target] + e2.cost) {
·               distance[e2.target] = distance[e1.target] + e2.cost;
·               queue.add(new Edge(e1.target, e2.target, distance[e2.target]));  // 追加
·           }
·       }
```

出力

```
for (int i = 0; i < vn; i++) {
        System.out.println(distance[i] == INF ? "INF"
    : distance[i]);

}
```

無向グラフの問題だったら

- edges.get(a).add(new Edge(a, b, f));
- edges.get(b).add(new Edge(b, a, f));

ワーシャルフロイド

- 全点対最短経路問題
- n ^ 3 (ノード数 100 とかまで)
- http://doriven.hatenablog.com/entry/2013/12/23/044921

解いてみよう

- 問題: All Pairs Shortest Path
- http://judge.u-aizu.ac.jp/onlinejudge/description.jsp?id=GRL_1_C&lang=jp

ワーシャルフロイドの流れ

- 2次配列初期化, d[頂点数][頂点数]
  - 自身のノード以外すべて最大コスト
  - 辺の登録
  - ワーシャルフロイド計算
  - -> 頂点同士の最短距離が求まる

入力受け取り

- int n = sc.nextInt(); // 頂点の数

- int en = sc.nextInt(); // 辺の数

WF 準備

```
int[][] d = new int[n][n];
for (int i = 0; i < n; i++) {
    Arrays.fill(d[i], INF);
    d[i][i] = 0;
}
```

入力受け取り

```
for (int i = 0; i < en; i++) {
    int s = sc.nextInt();
    int t = sc.nextInt();
    int dist = sc.nextInt();
    d[s][t] = dist;                    // 辺の追加
}
```

WF 実行

```
for (int k = 0; k < n; k++) {

    for (int i = 0; i < n; i++) {

        for (int j = 0; j < n; j++) {

            d[i][j] = Math.min(d[i][j],d[i][k]+d[k][j]);

        }}}
```

Negative Loop

- d[i][i] が 0 より小さい
  - -> コストが定まらないループに陥る
- d[i][i] (0 <= i < n) についてチェックする

出力

- d[from][to] で最短距離の結果が入っている
- 出力するだけ

追加資料

- 各アルゴリズムのAOJ 問題:
  - http://d.hatena.ne.jp/otaks/20111026/1319629788
- Visualizer:
  - http://jasonpark.me/AlgorithmVisualizer/#path=graph_search/dijkstra/shortest_path