最短路总结与次短路

主讲人：邓哲也

一点区分

- Dijkstra 算法、Bellman-Ford 算法、SPFA 算法是用来求解单源最短路径的。
- Floyd 是用来求多源最短路径的。

对比

算法	时间复杂度	特点	代码复杂度
Dijkstra 算法	O(n^2)	不能处理负权边 时间复杂度稳定	比较简单
Dijkstra 算法 + 堆优化	O((n+m) log n)		不简单
Bellman-Ford 算法	O(nm)	可以算带负权边的最短路，可以判断负环	简单
SPFA 算法	O(km)	Bellman-Ford 的队列优化版本	比较简单
Floyd 算法	O(n^3)	和邻接矩阵有一定联系	最简单

POJ 3255 Roadblocks

- 给定一个 n 个点，m 条边的无向图。
- 求出从 1 到 n 的严格次短路。（不能等于最短路，边可以重复经过）
- n <= 5000, m <= 10000
- 样例：　（输出 450）
- 4 4
- 1 2 100
- 2 4 200
- 2 3 250
- 3 4 100

POJ 3255 Roadblocks

- 首先，如果我们能算出次短路，那一定得知道最短路。
- 所以可以考虑在最短路算法上拓展一下，得到次短路的算法。

POJ 3255 Roadblocks

- dist[i] 表示从源点到 i 的最短路。
- 如果记一个 dist2[i] 表示从源点到 i 的严格次短路如何？

POJ 3255 Roadblocks

- 原本对于(u, v, w)是如何松弛的？
- if dist[u] + w < dist[v]
  - dist[v] = dist[u] + w

- 有了 dist2[] 之后该如何松弛？

POJ 3255 Roadblocks

```
if (dist[u] + w < dist[v]) {
    dist2[v] = dist[v];
    dist[v] = dist[u] + w;
} else if (dist[u] + w < dist2[v] && dist[u] + w != dist[v]){
    dist2[v] = dist[u] + w;
}
```

POJ 3255 Roadblocks

- 练习：用 dijkstra + 堆优化实现一下。

POJ 3255 Roadblocks

```
while (que.size() > 0) {
    int u = que.begin()->first;
    int dis = que.begin()->second;
    que.erase(que.begin());
    if (dist2[v] < dis) continue;
    for (int i = h[u]; i != -1; i = e[i].next) {
        int v = e[i].v;
        int d2 = dist[u] + e[i].w;;
        if (d2 < dist[v]) {
            dist2[v] = dist[v];
            dist[v] = d2;
            que.insert(make_pair(dist[v], v));
        } else if (d2 < dist2[v] && d2 != dist[v]) {
            dist2[v] = d2;
            que.insert(make_pair(dist2[v], v));
        }
    }
}
```

下节课再见