# 实验 6 图的应用（最小花费）

计科 2004 202008010404 赖业智

## 一、问题分析

1. 问题与功能描述

1）需要处理的是一组 int 型数字，它们共同组成了图的元素（顶点、边、边权）。

2）实现的功能有，输入两个正整数 n,m，分别表示总人数和可以互相转账的人的对数，然后循环 m 次，每次输入三个正整数 x,y,z，表示标号为 x 的人和标号为 y 的人之间互相转账需要扣除 z%的手续费(z<100)。利用 dijkstra 算法得出单源最短路径 (顺为最长，逆为最短）。

3）使用标准输入输出，且结果保留八位小数，无须四舍五入。

2. 样例分析

1）输入样例

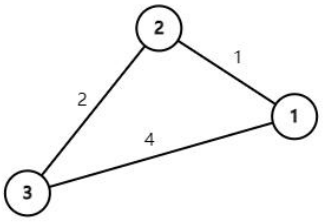
```
3 3
1 2 1
2 3 2
1 3 4
1 3
```

2）输出样例

```
103.07153164
```

3）样例说明

我们利用这些数据来构造一个无向图，3 是该图的顶点，3 也是该图的边数，由于可以互相转账，因此是无向图。图如下：




把边值转化为扣完手续费的值比较容易理解，设原来至少为 x，即 1 到 2 后 0.99x，2 到 3 后 0.98*0.99x=100，这样来看，是求最长路，如果不转化，还是一个最短路问题。因此最后 x 为 103.07153164。

## 二、数据结构分析

1. 数据对象：本题处理 int 型数组

2. 数据关系：本题处理的数据为整型有限数据，题目每组数据之间均可能有关系。因此使用图作为基本数据结构进行操作。

3. 基本操作

【功能描述】设置图的边

【名字】setEdge

【输入】顶点 1，顶点 2，边权

【输出】无

【功能描述】求解未加入路径的目前对应的最大权值结点

【名字】maxVertex

【输入】无

【输出】该顶点的下标

【功能描述】求解最优路径

【名字】Dijkstra1

【输入】double D[G->n()]和起始顶点的下标

【输出】无

4.物理实现

```
void setEdge(int v1, int v2, double wt)
    {
//    Assert(wt>0, "Illegal weight value");
        assert(wt > 0);
        if (matrix[v1][v2] == 0) numEdge++;
        matrix[v1][v2] = wt;
     }

void Dijkstra1(double* D, int s)
    {
      int i,v,w;
      for(i=0;i<G->n();i++){
            v=maxVertex(D);
            if(D[v]==-1) return;
            G->setMark(v,VISITED);
            for(w=G->first(v);w<G->n();w=G->next(v,w)){
                  if(D[w]>1.0) D[w]=0.0;
                  if(D[v]==0) D[v]=1.0;
                        double x=1.0*D[w];
                  double y=1.0*(D[v]*G->weight(v,w));
            if(x<y)
                  D[w]=y;//顺找最长路(0.98 和 0.99 比，选 0.99)
            }
            }
```

```
    }

int maxVertex(double* D)    // Find min cost vertex
    {
        int i, v = -1;
        // Initialize v to some unvisited vertex
        for (i = 0; i < G->n(); i++)
            if (G->getMark(i) == UNVISITED)
            {
                v = i;
                break;
            }
        for (i++; i < G->n(); i++) // Now find smallest D value
            if ((G->getMark(i) == UNVISITED) && (D[i] > D[v]))
                v = i;
        return v;
    }
```

# 三、算法分析

1.算法思想：声明一个数组 dis 来保存源点到各个顶点的最长距离和一个保存已经找到了最长路径的顶点的集合 T。初始时，原点 s 的路径权重被赋为 0 （dis[s] = 0）。若对于顶点 s 存在能直接到达的边（s,m），则把 dis[m]设为 w（s, m）,同时把所有其他（s 不能直接到达的）顶点的路径长度设为 0。初始时，集合 T 只有顶点 s。然后，从 dis 数组选择最大值，则该值就是源点 s 到该值对应的顶点的最长路径，并且把该点加入到 T 中，此时完成一个顶点，接着，我们需要看看新加入的顶点是否可以到达其他顶点并且看看通过该顶点到达其他点的路径长度是否比源点直接到达长，如果是，那么就替换这些顶点在 dis 中的值。最后，又从 dis 中找出最大值，重复上述动作，直到 T 中包含了图的所有顶点。

3. 性能分析：

【空间复杂度】对于 Dijkstra 算法仅仅开辟了一个数组来存储最长距离，因此空间复杂度为 O(n)。

【时间复杂度】Dijkstra 算法对应的两个函数各有两个 for 循环，一个嵌套，一个并列。相

当于在外层 for 循环的内部加入了三个并列的 for 循环，即时间复杂度为 O(n*n)。