链滴

PAT 甲级刷题实录——1003

作者：aopstudio
原文链接：https://ld246.com/article/1578664126630
来源网站：链滴

原题链接

https://pintia.cn/problem-sets/994805342720868352/problems/994805523835109376

思路

这题基本上就是采用迪杰斯特拉算法求最短路径。只是我一开始给想复杂化了，结果走了不少弯路。为城市的数量最多可以500个，是个相当大的数，同时并不是每个城市之间都有路，如果用矩阵存储的话是个稀疏矩阵，因此我就想用邻接表存。这下可好，光是读入数据到邻接表就写了我80行代码，后的算法更是无从下手了，完全走进死胡同了，无奈之下，只好去网上查人家的做法，发现人家压根没考虑稀疏矩阵的事，都是用邻接矩阵存，更有甚者直接定义了一个500*500的矩阵，空间复杂度高了极致。我还是折中一下吧，用题目中给的城市总数定义邻接矩阵大小，实现方法还是我们最喜欢的vctor，为此我特意搜索了一下vector实现二维数组的方法，应该是这样的vector<vector<int> > matrx(n, vector<int>(m))，注意两个>之间有空格。这道题需要在传统迪杰斯特拉算法上增加的东西是：1 权值相同的最短路径总数。2. 可以集合到的救援队总数。这些在代码里都有体现，应该一看就能明。

代码

```
#include <iostream>
#include <vector>
#include <limits.h>
using namespace std;

int main()
{
    int cityNum, roadNum, depa, dest;
    cin >> cityNum >> roadNum >> depa >> dest;
    vector<vector<int> > adjMatrix(cityNum, vector<int>(cityNum,INT_MAX));    //邻接矩阵
    vector<int> dist(cityNum,INT_MAX);    //记录到各个城市的最短距离
    vector<int> pathSum(cityNum);    //到各个城市的最短路径数量
    vector<int> teamSum(cityNum);    //到各个城市能集合到的救援队数量
    vector<bool> reach(cityNum);    //记录计算过程中是否以最短路径到各个城市
    vector<int> teams(cityNum);    //每个城市救援队数量
    for (int i = 0; i < cityNum; i++)
    {
        adjMatrix[i][i] = 0;
    }
    for (int i = 0; i < cityNum; i++)
    {
        int tnum;
        cin >> tnum;
        teams[i]=tnum;
    }
    for (int i = 0; i < roadNum; i++)
    {
        int from, to, weight;
        cin >> from >> to >> weight;
        adjMatrix[from][to] = weight;
        adjMatrix[to][from] = weight;
        if (from == depa)    //根据出发点的相邻路径更新相关信息
        {
```

```
            dist[to] = weight;
            teamSum[to] = teams[from] + teams[to];
            pathSum[to] = 1;
        }
    }
    dist[depa] = 0;
    teamSum[depa] = teams[depa];
    pathSum[depa] = 1;
    reach[depa] = true;
    //注意，只有当reach[i]为true时，dist[i]才是出发点到它真正的最短路径长度，否则就还得更新
    for (int i = 0; i < cityNum; i++)
    {
        int min = INT_MAX, u=-1;
        for (int j = 0; j < cityNum; j++)
        {
            if (reach[j] == false && dist[j] < min)
            {
                min = dist[j];
                u = j;
            }
        }
        if (min == INT_MAX)
            break;
        reach[u] = true;
        for (int j = 0; j < cityNum; j++)
        {
            if (reach[j] == false && adjMatrix[u][j]!=INT_MAX && dist[u] + adjMatrix[u][j] < dist[j])

                //若adjMatrix[u][j]==INT_MAX就直接排除掉，如果没有判断，那么当它为INT_MAX的
候，dist[u]+adjMatrix[u][j]会反转变成负数，这样反而会符合<dist[j]的要求
            {
                dist[j] = dist[u] + adjMatrix[u][j];    //更新最短距离
                pathSum[j] = pathSum[u];    //更新可达的最短路径总数
                teamSum[j] = teamSum[u] + teams[j];        //更新可以集合到的救援队总数
            }
            else if (reach[j] == false && (dist[u] + adjMatrix[u][j]) == dist[j])    //如果有另一条最短
离相同的路径
            {
                pathSum[j] += pathSum[u];    //增加可达的最短路径总数
                if (teamSum[j] < teamSum[u] + teams[j])    //如果这条路径可以集合到的救援队数量更

                    teamSum[j] = teamSum[u] + teams[j];    //更新可以集合到的救援队数量
            }
        }
    }
    cout << pathSum[dest] << ' ' << teamSum[dest];
}
```

感想和注意事项

1. 当节省空间的方法过于复杂时，尽量用空间换时间，而不是想着怎么去节省空间。比如说图用邻接阵而不是邻接表存，这样读入数据以及之后的运算过程都会快很多。

2. 代码中用到了一个INT_MAX值，表示不可达的路径长度，这个常量包含在limits.h头文件中。但是

要注意的是，如果把INT_MAX的值加上一个正数以后，会反转变成负数，从而导致程序出现错误。此我在代码中进行了判断。我看网上有些方法就是自己定义一个常量INF，再给它手工赋一个很大但不超过int表示范围的数比如1111111，从而来表示不可达的路径长度，这样也行，但我个人感觉不太谨。

3. vector真是神器，太好用了。当然你也可以直接定义一个静态数组，用一个很大的数表示它的容量事实上网上大部分解答都是这么做的，但我不用这种方法也是和上面一样的原因，这种做法太不严谨且太浪费空间了。当能够用很低的代价（多几行代码）节省大量的空间时，这种做法还是值得的。