华东师范大学数据科学与工程学院上机实践报告

课程名称：算法设计与分析　　**年级：**21 级　　**上机实践成绩：**

指导教师：金澈清　　**姓名：**杨茜雅

上机实践名称：并查集　　**学号：**10215501435　　**上机实践日期：**

上机实践编号：No.13

一、目的

1．熟悉经典单源最短路径算法

二、内容与设计思想

1. 解决 OJ 上的练习题，习题描述如下所述。
2. 使用 Bellman-Ford 算法和 Dijkstra 算法。
3. 尝试使用优先队列对 Dijkstra 算法进行优化。
4. 总结以上算法的时间复杂度、优缺点以及使用场景。

题目：

某国有 n 个城市和 m 条单向火车，可能存在重边和自环，所有火车费用均为正值。(1-1000)

请你求出 1 号城市到 n 号城市的最少费用，如果无法从 1 号城市走到 n 号城市，则输出−1。

输入：

第一行包含整数 n 和 m。(1<=n<=500, 1<=m<100000)

接下来 m 行每行包含三个整数 x,y,z，表示存在一条从城市 x 到城市 y 的火车，价格为 z。

输出：

输出一个整数，表示 1 号城市到 n 号城市的最少费用。

如果路径不存在，则输出−1。

三、使用环境

推荐使用 C/C++集成编译环境。

不建议随意改变本文档以上部分的结构，以下部分可以按需扩充。但不要缺少“实验过程”和“总结”。

四、实验过程

Dijkstra（优化前）

```
#include<iostream>
#include<cstring>
#include<algorithm>

using namespace std;

const int N=520;

int n,m;
int g[N][N];
int dist[N];
bool st[N];

int dijkstra()
{
    memset(dist,0x3f,sizeof dist);
    dist[1]=0;
    for(int i=0;i<n-1;i++)
    {
        int t=-1;
        for(int j=1;j<=n;j++)
        {
            if(!st[j]&&(t==-1||dist[t]>dist[j]))
            t=j;
        }
        for(int j=1;j<=n;j++)
            dist[j]=min(dist[j],dist[t]+g[t][j]);
            st[t]=true;
    }
    if(dist[n]==0x3f3f3f3f)
    return -1;
    else return dist[n];
}
int main()
```

```
{
    memset(g,0x3f,sizeof g);
    cin>>n>>m;
    for(int i=1;i<=m;i++)
    {
        int a,b,c;
        cin>>a>>b>>c;
        g[a][b]=min(g[a][b],c);
    }
    cout<<dijkstra()<<endl;
    return 0;
}
```

堆优化

```
#include<iostream>
#include<queue>
#include<string.h>
#include<vector>
using namespace std;
int m,n;
const int N=100010;
int d[N];
int ne[N],e[N],idx,h[N],w[N];
bool st[N];
typedef  pair<int,int>   P;
void add(int a,int b,int c)
{
    e[idx]=b;
    ne[idx]=h[a];
    w[idx]=c;
    h[a]=idx++;
}
int djs()
{
```

```
    memset(d,0x3f,sizeof(d));
    d[1]=0;
    priority_queue<P,vector<P>,greater<P>>q;
        q.push({0,1});
    while(q.size())
    {
       auto t=q.top();
        int dis=t.first,ver=t.second;
        q.pop();
        if(st[ver])
        continue;
        st[ver]=1;
        for(int i=h[ver];i!=-1;i=ne[i])
        {
            int j=e[i];
            if(d[j]>dis+w[i])
            {
                d[j]=dis+w[i];
                q.push({d[j],j});
            }
        }
    }
    if(d[n]==0x3f3f3f3f)
    return -1;
    return d[n];
}
int main()
{
      memset(h,-1,sizeof(h));
     cin>>n>>m;
       while(m--)
    {
         int a,b,c;
       cin>>a>>b>>c;
```

```
        add(a,b,c);
    }
     int t= djs();
    printf("%d\n",t);
   return 0;
}
```

五、总结

Bellman-ford 算法复杂度为 O（VE），有负权重也可以适用，但是每一次都会更新所有节点，效率不高。Dijkstra 算法只适用于所有权重为正的情况，每次更新节点的个数会随着 S 集合里节点的个数增多而越来越小，算法时间复杂度为 O (V) + O (V^2) + O (V N) ~ O (V^2)。

基于优先队列的算法节约了每次寻找距离最小顶点的开销 O (V)，然而增加了每次更新邻居的开销 O (logV)，总复杂度为 O (V N log V)