# 实验 4-最短路径

PB17081504
廖洲洲

## 1. 实验要求

1）输入交通网 G

2）用 Dijkstra 算法计算从始发站到终点站的最短路径

## 2. 实验内容

1）定义

- Dijkstra 算法是典型的单源最短路径算法，用于计算一个节点到其他所有节点的最短路径。主要特点是以起始点为中心向外层层扩展，直到扩展到终点为止。

2）算法描述

- 算法思想：设 G=(V,E)是一个带权有向图，把图中顶点集合 V 分成两组，第一组为已求出最短路径的顶点集合（用 S 表示，初始时 S 中只有一个源点，以后每求得一条最短路径 ，就将加入到集合 S 中，直到全部顶点都加入到 S 中，算法就结束了），第二组为其余未确定最短路径的顶点集合（用 U 表示），按最短路径长度的递增次序依次把第二组的顶点加入 S 中。在加入的过程中，总保持从源点 v 到 S 中各顶点的最短路径长度不大于从源点 v 到 U 中任何顶点的最短路径长度。此外，每个顶点

对应一个距离，S 中的顶点的距离就是从 v 到此顶点的最短路径长度，U 中的顶点的距离，是从 v 到此顶点只包括 S 中的顶点为中间顶点的当前最短路径长度。

3）算法步骤：

a）初始时，S 只包含源点，即 S＝{v}，v 的距离为 0。U 包含除 v 外的其他顶点，即:U={其余顶点}，若 v 与 U 中顶点 u 有边，则<u,v>正常有权值，若 u 不是 v 的出边邻接点，则<u,v>权值为∞。（本实验用-1 来代替）

b）从 U 中选取一个距离 v 最小的顶点 k，把 k，加入 S 中（该选定的距离就是 v 到 k 的最短路径长度）。

c）以 k 为新考虑的中间点，修改 U 中各顶点的距离；若从源点 v 到顶点 u 的距离（经过顶点 k）比原来距离（不经过顶点 k）短，则修改顶点 u 的距离值，修改后的距离值的顶点 k 的距离加上边上的权。

d）重复步骤 b 和 c 直到所有顶点都包含在 S 中。

## 3. 实验关键代码讲述

```
void ShortestPath(){
    int *final,*D;
    int **path;
    final=(int *)malloc(vexnum*sizeof(int));
//final[i]为 1，说明已求得从 startvex 到 vi 的最短路径
    D=(int *)malloc(vexnum*sizeof(int));
//D[i]中存储了从 startvex 到 vi 的路径长度

    path=(int **)malloc(vexnum*sizeof(int *));
    for(int i=0;i<vexnum;i++)
```

```
        path[i]=(int *)malloc(vexnum*sizeof(int));
//path[v]存储了从起点到 v 的最短路径

    for(int i=0;i<vexnum;i++){
        final[i]=0;
        if(i>startvex)
            D[i]=arc[startvex][i];
        else
            D[i]=arc[i][startvex];
        for(int w=1;w<vexnum;w++)
            path[i][w]=-1;
        path[i][0]=startvex;
    }//for
    int w,v,min;int j;
    D[startvex]=0;final[startvex]=1;
//开始时，vo 属于 S
    int flag=0;//表示未找到到终点得路径
//每次求得 startvex 到某个 v 顶点的最短路径，并加入 S 集
    for(int i=1;i<vexnum&&flag==0;i++){
        min = max;
        for(w=0;w<vexnum;w++){
//找出当前所知 S 中离起点最近的点
            if(!final[w])
                if(D[w]>0&&D[w]<min){
                    v=w;
                    min=D[w];
                }
        }
    final[v]=1;
    for( j=0;path[v][j]!=-1;j++);
    path[v][j]=v;
    int m,n;
    for(w=0;w<vexnum;w++){
//更新最短路径
        if(v<w){
            m=v;n=w;
        }
        else{
            m=w;n=v;
        }
        if(!final[w]&&arc[m][n]>0&& ( (min+arc[m][n])<D[w] || D[w]<0 ) )
{  //如果发现有更短的路径长度，修改 D[w]
            D[w]=min+arc[m][n];
            for(int j=0;path[v][j]!=-1;j++)
```

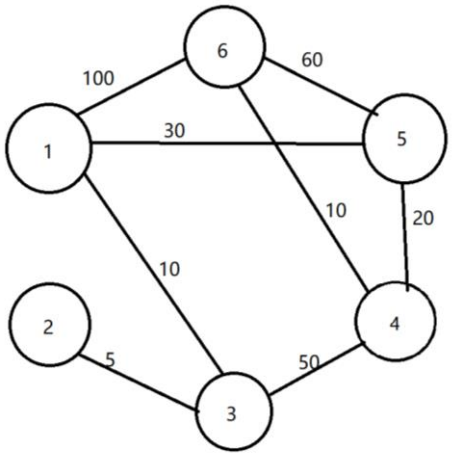
```
                path[w][j]=path[v][j];
        }

    }
    if(v==endvex)
        flag=1;
    }//路径输出
    printf("最短路径查找完成！\n");
    printf("长度为：%d\n",D[endvex]);
    printf("路径为：\n");
    for(int j=0;path[endvex][j]!=-1;j++)
        printf("%d  ",path[endvex][j]+1);

}
```

## 4. 实验结果及分析

- 输入如图路径




0 -1 10 -1 30 100

0 0 5 -1 -1 -1

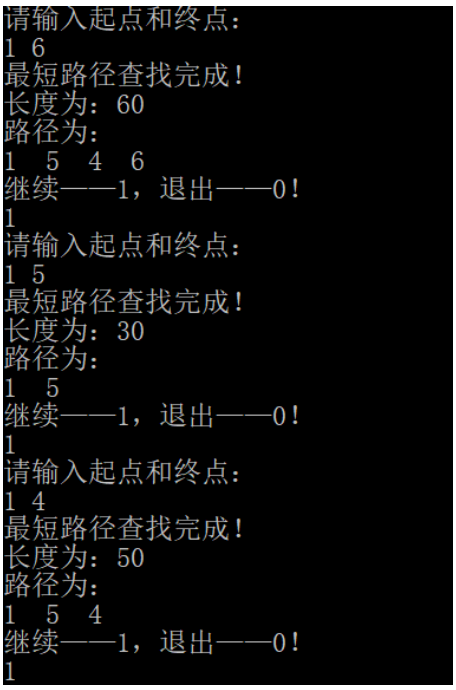
0 0 0 50 -1 -1

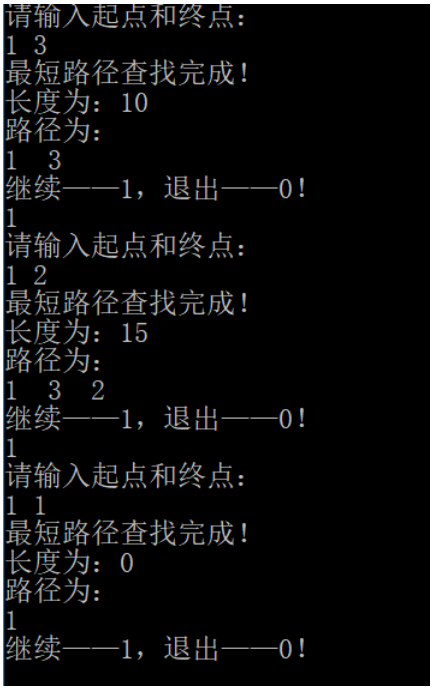
0 0 0 0 20 10

0 0 0 0 0 60

0 0 0 0 0 0

- 输入起点、终点




可以发现实验结果完全正确！

## 5. 实验小结

- 通过本次实验对图的存储、基本操作有了更加深刻的了解
- 学习了 Dijkstra 算法，明白了最短路径的求法