

## 1.题目名称

社交网络图中结点的“重要性”计算

## 2.代码行数

95行

## 3.算法思想

### Dijkstra

利用Dijkstra算法找到题目要求点到每个点的最短路径即可。

## 4.主要/核心函数分析

### Dijkstra

```
1
2 void Dijkstra(int begin){
3     for(int i=1;i<=n;i++){           //初始化dist
4         dist[i].value=Matrix[begin][i];
5     }
6     dist[begin].value=0;
7     dist[begin].visit= true;
8     int count=1,temp,min;
9     while(count!=n){
10        temp=1;
11        min=INF;
12        for(int i=1;i<=n;i++){       //寻找value最小且为遍历的点
13            if(!dist[i].visit && dist[i].value<min){
14                min=dist[i].value;
15                temp=i;
16            }
17        }
18
19        dist[temp].visit= true;
20        count++;
21        for(int i=1;i<=n;i++){       //遍历一下寻找最短路径
22            if(!dist[i].visit && Matrix[temp][i]!=INF &&
23                (dist[temp].value+Matrix[temp][i])<dist[i].value){
24                dist[i].value=dist[temp].value+Matrix[temp][i];
25            }
26        }
27    }
```

#### 1. 函数流程:

- 初始化:
  - 初始化 `dist` 数组, 将所有距离设置为 `Matrix` 中对应的值。
  - 将起始节点 `begin` 的距离设置为0。
  - 将起始节点标记为已访问。
- 主循环:

- 当未访问的节点数量不为n时，继续执行以下步骤。
- 寻找未访问的节点中距离最短的节点。
- 将该节点标记为已访问。
- 遍历所有节点，更新与该节点直接相连的未访问节点的距离。

## 5.测试数据(规模,测试次数)

规模: $N \leq 1000$  &  $M \leq 10000$  &  $K \leq 100$

测试次数:1

测试用例:见测试文件

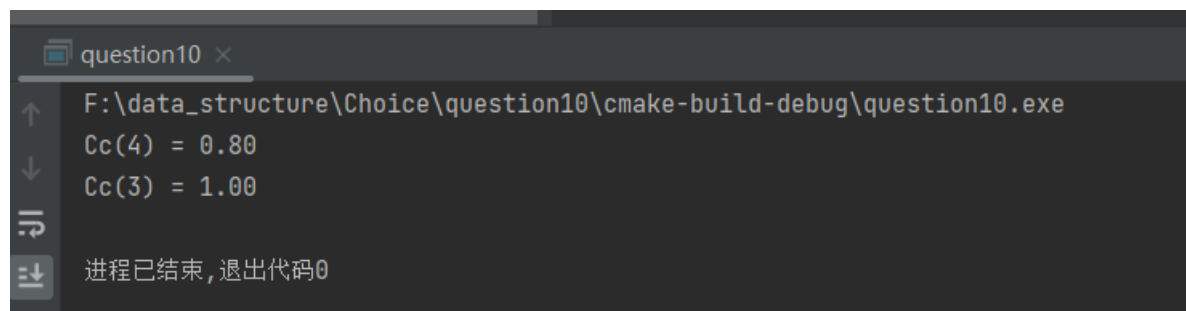
## 6.运行结果

```
1 F:\data_structure\Choice\question10\cmake-build-debug\question10.exe
2 Cc(4) = 0.80
3 Cc(3) = 1.00
4
5 进程已结束,退出代码0
6
```

## 7.时间复杂度分析

代码使用Dijkstra算法，因此时间复杂度是 $O((V+E)\log V)$ ，其中V是顶点的数量，E是边的数量。

## 8.结果截屏图片



```
question10 x
F:\data_structure\Choice\question10\cmake-build-debug\question10.exe
Cc(4) = 0.80
Cc(3) = 1.00
进程已结束,退出代码0
```

## 9.心得体会

对于Dijkstra算法求目标点到每个点的最短路径更加熟悉。