1.题目名称

最小生成树

2.代码行数

161行

3.算法思想

分别调用Krustal算法和Prim算法求最小生成树以及它的权值即可。

4.主要/核心函数分析

Prim

```
void Prim(){
    dist[0]=0;
    visit[0]= true;
    path[0]="v1";
    for(int i=1;i<n;i++){
        dist[i]= min(dist[i],Matrix[0][i]);
        path[i]= "v"+to_string(1)+"->"+"v"+ to_string(i+1);
    }
    cout<<"v"<<1<<"加入顶点集合"<<endl;
    cout<<path[0]<<endl;
    Print();
    for(int i=1;i<n;i++){
        int temp=-1;
        int min=INF;
        for(int j=1;j<n;j++){
            if(!visit[j]&&dist[j]<min){
                min=dist[j];
                temp=j;
            }
        }
        if(temp==-1){
            flag= true;
            return;
        }

        visit[temp]= true;
        sum+=dist[temp];
        for(int j=1;j<n;j++){
            if (dist[j]>Matrix[temp][j]){
                dist[j]=Matrix[temp][j];
                path[j]=path[temp]+"->"+"v"+ to_string(j+1);
            }
        }
        cout<<"v"<<temp+1<<"加入顶点集合"<<endl;
        cout<<path[temp]<<endl;
        Print();
    }
}
```

```
39
```

1. 从源点出发，将所有与源点连接的点加入一个待处理的集合中
2. 从集合中找出与源点的边中权重最小的点，从待处理的集合中移除标记为确定的点
3. 将找到的点按照步骤1的方式处理
4. 重复2，3步直到所有的点都被标记

Krustal

```
void Krustal(){
    Initial();
    for(int i=0;i<m;i++){
        int x= find(e[i].u);
        int y= find(e[i].v);
        if(x!=y){
            cout<<e[i].u<<"->"<<e[i].v<<endl;
            if(Union(x,y)){
                sum+=e[i].w;
            }
        }
    }
}
```

对边的权值按从小到大排。

从小到大依次进行插入，如果边关联的两个点属于同一个树则直接跳过不执行插入。

5.测试数据(规模,测试次数)

规模:边的权值为float类型，运算结果不超过float类型最大值

测试次数:1

测试用例:见测试文件

6.运行结果

```
F:\data_structure\Choice\question22\cmake-build-debug\question22.exe
v1加入顶点集合
v1
v1  v2  v3  v4  v5  v6  v7  v8  v9  v10
0   3.7 2   INF INF INF INF INF INF INF

v3加入顶点集合
v1->v3
v1  v2  v3  v4  v5  v6  v7  v8  v9  v10
0   3.7 2   INF INF 4   6   INF INF INF

v2加入顶点集合
v1->v2
v1  v2  v3  v4  v5  v6  v7  v8  v9  v10
0   3.7 2   4   5.5 4   6   INF INF INF

v4加入顶点集合
v1->v2->v4
v1  v2  v3  v4  v5  v6  v7  v8  v9  v10
0   3.7 2   4   5.5 4   6   6   INF 8
```

```
v6加入顶点集合
v1->v3->v6
v1  v2  v3  v4  v5  v6  v7  v8  v9  v10
0   3.7 2   4   5.5 4   1.2 2   4   8

v7加入顶点集合
v1->v3->v6->v7
v1  v2  v3  v4  v5  v6  v7  v8  v9  v10
0   3.7 2   4   5.5 1.2 1.2 2   4   8

v8加入顶点集合
v1->v3->v6->v8
v1  v2  v3  v4  v5  v6  v7  v8  v9  v10
0   3.7 2   4   3   1.2 1.2 2   4   4.4

v5加入顶点集合
v1->v3->v6->v8->v5
v1  v2  v3  v4  v5  v6  v7  v8  v9  v10
0   3.7 2   4   3   1.2 1.2 2   4   4.4

v9加入顶点集合
v1->v3->v6->v9
v1  v2  v3  v4  v5  v6  v7  v8  v9  v10
0   3.7 2   4   3   1.2 1.2 2   4   4.4

v10加入顶点集合
v1->v3->v6->v8->v10
v1  v2  v3  v4  v5  v6  v7  v8  v9  v10
0   3.7 2   4   3   1.2 1.2 2   4   4.4

ALL:28.3

6->7
1->3
6->8
5->8
1->2
2->4
6->9
3->6
8->10
ALL:28.3

进程已结束,退出代码0
```

7.时间复杂度分析

Prim算法时间复杂度为$O(n^2)$

Krustal算法的时间复杂度为O(nlogn)

8.结果截屏图片

9.心得体会

对于两种求最小生成树的算法更加了解与熟悉。同时也明白了两种算法的时间复杂度优劣。