西安交通大学实验报告

课程名称： 算法设计与问题求解 实验名称： 最小生成树和Kruskal算法

学 院： 机械工程学院 实验日期 2020 年 11 月 28 日

班 级： 机械97班 姓 名： 杨逢诜 学号： 2193712613

#### 一、实验内容和结果

* **题目1 ：实验：克鲁斯卡尔算法编程**

采用C++语法，利用Kruskal算法计算出指定无向赋权图的最小生成树并指出其涉及到的点和边。

【源程序】

#include<stdio.h> //本次作业中由于使用了C++语言特有的布尔变量，因此需要引用iostream头函数和std命名空间

#include<iostream>

using namespace std;

int MAX=100; //定义最大值

typedef struct //定义边结构体，包含边的起点、重点、权重以及“是否包含在最小生成树内”

{

int begin;

int end;

int weight;

bool selection;

//int selection;（这个是完全用C语言时的老写法）

}Edge;

void Kruskal(Edge\* edge,int n,int m,int\* parent) //声明函数，用于执行Kruskal算法计算最小生成树

{

int i,j; //声明循环控制变量

for(i=0;i<m-1;i++) //排序算法：将全部边按权值由小到大排序

{

for(j=i+1;j<m;j++) //这里采用选择排序算法：“如果它后面有权重比我小的边，那么就让这两条边交换次序”

{

if(edge[i].weight>edge[j].weight)

{

Edge k=edge[i];

edge[i]=edge[j];

edge[j]=k;

}

}

}

i=0,j=0; //重置循环控制变量，开始构建最小生成树

while(j<n-1) //采用WHILE语句，j用于控制最小生成树中边的总数

{

int p=edge[i].begin,q=edge[i].end; //声明两个辅助变量，用于寻找同一边对应两点在生成过程中的根

while(parent[p]!=-1) //寻找边起点的最深层次的根（下同，但是被操作的是边的终点）

{

p=parent[p];

}

while(parent[q]!=-1)

{

q=parent[q];

}

if(p!=q) //若边的起讫点具有不同的根，那么依据Kruskal算法，这条边将包含在最小生成树中

{

edge[i].selection=true;

parent[edge[i].end]=edge[i].begin; //将该边的讫点定义为“起点的子结点”

j++;

}

i++; //对这一条边的运算结束，开始对下一条边进行运算

}

}

int main() //主函数开始

{

int parent[100],n,m,i,j; //声明整型数组，用于描述在生成树中结点的父节点；声明整型变量，用于存储图的总体信息

Edge edges[100]; //声明结构体数组，用于描述图中全部边的信息

printf("请输入图中包含的总点数和总边数:\n"); //根据提示输入图中包含的边、点数目并将其存入内存中；

scanf("%d %d",&n,&m);

//cout<<"请输入图中包含的总点数和总边数："<<endl;

//cin>>n>>m;

printf("请依次输入图中包含的边的起始点和权值:\n"); //根据提示输入图中全部边的具体信息并将其存入内存中；

//cout<<"请依次输入图中包含的边的起始点和权值："<<endl;

for(i=0;i<m;i++)

{

scanf("%d %d %d",&edges[i].begin,&edges[i].end,&edges[i].weight);

//cin>>edge[i].begin>>edge[i].end>>edge[i].weight;

//edge[i].selection=false;

edges[i].selection=false; //初始化边的信息：所有边在经历算法运算前都未被选中，在生成树中都不具有父节点

parent[i]=-1;

}

Kruskal(edges,n,m,parent); //调用Kruskal函数，找出最小生成树中边的数量

j=1;

for(i=0;i<m;i++) //根据要求输出函数运算结果，主函数结束

{

if(edges[i].selection==true)

{

printf("被选中的第%d条边的起始点是%d和%d，权值是%d\n",j,edges[i].begin,edges[i].end,edges[i].weight);

j++;

}

}

return 0;

}

【运行结果】

