数据结构上机报告

题目：最小生成树

班级：04班 姓名： 学号： 完成日期：2020.12.06

一、需求分析

1、问题描述

若要在n个城市之间建设通讯网络，只需要架设n-1条线路即可。如何以最低的经济代价建设这个通讯网，是一个网的最小生成树问题。

2、基本要求

（1）利用克鲁斯卡尔算法求网的最小生成树。

（2）实现并查集。以此表示构造生成树过程中的连通分量。

（3）以文本形式输出生成树中各条边以及他们的权值。程序功能

3、预设条件

通讯线路一旦建立，必然是双向的。因此，构造最小生成树的网一定是无向网。设图的顶点数不超过30个，并为简单起见，网中边的权值设成小于100的整数，可利用C语言提供的随机数函数产生。

图的存储结构的选取应和所作操作向适应。为了便于选择权值最小的边，此题的存储结构既不选用邻接矩阵的数组表示法，也不选用邻接表，而是以存储边(带权)的数组表示图。

二、概要设计:

将输入的带权边依照权值大小进行排序，并初始化根节点数组。然后用克鲁斯卡尔算法求出最小生成树并输出。

图带权边实现：

定义一个结构体类型表示带权边

typedef struct { //边的数据结构

int begin;

int end;

int weight;

3、主程序流程

a.将给定的数据读入到带权边数组中，并将带权边数组依照权值排序。

b.利用查找函数和克鲁斯卡尔算法求得最小生成树并输出。

scanf("%d %d",&n,&m); //输入结点数和边数

Edge a[m];

int parent[n+1];

for(i = 0;i < m;i++) //按边输入图

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

for(i = 0;i < n+1;i++) //根节点初始化

parent[i] = 0;

sort(a,m); //按照边的权值排序

kruskal(a,parent,m); //生成并按边输出最小生成树

三、详细设计

实现伪码

带权边排序函数：

void sort(Edge \*a,int n) { //图中边的排序算法（选择排序）

int i,j;

for(i = 0;i < n;i++) {

for(j = i + 1;j < n;j++) {

if(a[i].weight > a[j].weight) {

int t1 = a[i].begin,t2 = a[i].end,t3= a[i].weight;

a[i].begin = a[j].begin;

a[i].weight = a[j].weight;

a[i].end = a[j].end;

a[j].begin = t1;

a[j].end = t2;

a[j].weight = t3;

}

}

}

}

查找函数：

//MFSet

int Find(int \*parent,int f) { //查找函数， 用于定位最小生成树的根节点

while(parent[f]) {

f = parent[f];

}

return f;

}

int max(int a,int b,int type) { //求两个数中的较大/较小值

if(a > b) {

if(type) return a;

else return b;

}

if(type) return b;

else return a;

}

克鲁斯卡尔算法：

void kruskal(Edge \*a,int parent[],int m) { //克鲁斯卡尔算法

int i,begin,end;

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

begin = Find(parent,a[i].begin);

end = Find(parent,a[i].end);

if(begin != end) {

parent[begin] = end; //merge函数，实现归并操作

printf("%d %d %d\n",max(a[i].begin,a[i].end,0),max(a[i].begin,a[i].end,1),a[i].weight);

}

}

}

四、调试分析

1、调试问题

a.初期实现程序时，在进行排序过程中，只进行了对应边的权值交换，而未实现对应结点的交换，导致程序错误，后经过改进排序算法实现了问题。

2、算法的时空复杂度分析和改进设想

算法的时间复杂度为O（m^2），m为对应边数。因为排序算法采用了速度较慢的选择排序，目前存在空间复杂度较高的问题，后期通过排序函数，将排序方式改为快速排序，可以减少循环的次数，降低算法的时间复杂度。

3、经验和体会

对克鲁斯卡尔算法有了更深的了解，克鲁斯卡尔算法属于贪心算法，具有局部最优解的特性。但在最小生成树问题中则实现了全局最优。

五、用户使用说明

用户只需运行所给文件夹下exe程序，按照给定的顺序输入结点、边数及带权边，即可得到输岀结果。

六、测试结果

输入样例

8 11

1 2 3

1 4 5

1 6 18

2 4 7

2 5 6

3 5 10

3 8 20

4 6 15

4 7 11

5 7 8

5 8 12

输出样例

1 2 3

1 4 5

2 5 6

5 7 8

3 5 10

5 8 12

4 6 15

1. 附录

#include<stdio.h>

typedef struct { //边的数据结构

int begin;

int end;

int weight;

}Edge;

void sort(Edge \*a,int n) { //图中边的排序算法（选择排序）

int i,j;

for(i = 0;i < n;i++) {

for(j = i + 1;j < n;j++) {

if(a[i].weight > a[j].weight) {

int t1 = a[i].begin,t2 = a[i].end,t3= a[i].weight;

a[i].begin = a[j].begin;

a[i].weight = a[j].weight;

a[i].end = a[j].end;

a[j].begin = t1;

a[j].end = t2;

a[j].weight = t3;

}

}

}

}

//MFSet

int Find(int \*parent,int f) { //查找函数， 用于定位最小生成树的根节点

while(parent[f]) {

f = parent[f];

}

return f;

}

int max(int a,int b,int type) { //求两个数中的较大/较小值

if(a > b) {

if(type) return a;

else return b;

}

if(type) return b;

else return a;

}

void kruskal(Edge \*a,int parent[],int m) { //克鲁斯卡尔算法

int i,begin,end;

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

begin = Find(parent,a[i].begin);

end = Find(parent,a[i].end);

if(begin != end) {

parent[begin] = end; //merge函数，实现归并操作

printf("%d %d %d\n",max(a[i].begin,a[i].end,0),max(a[i].begin,a[i].end,1),a[i].weight);

}

}

}

int main() {

int i,m,n;

scanf("%d %d",&n,&m); //输入结点数和边数

Edge a[m];

int parent[n+1];

for(i = 0;i < m;i++) //按边输入图

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

for(i = 0;i < n+1;i++) //根节点初始化

parent[i] = 0;

sort(a,m); //按照边的权值排序

kruskal(a,parent,m);

return 0;

}