# 离散数学实验 2

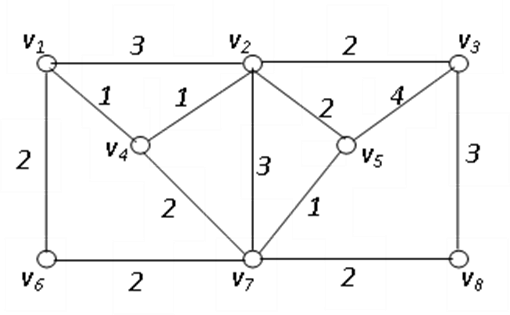
**根据上述思路实现 Kruskal 算法，提示：** 用如下结构体表示边：

struct Edge

{ int v1, v2; //边的两个顶点int weight; //边的权值

};

测试用例：



图中顶点数为 8，边数为 13，边集可以直接排序后在程序中初始化（已按权值排序）：{(1 4 1)，(2 4 1)，(5 7 1)，(1 6 2)， (2 3 2)，(2 5 2)，(4 7 2)，(6 7 2)，(7 8

2)，(1 2 3)，(2 7 3)，(3 8 3)，(3 5 4)}。（每个三元组表示一条边，其中前两个数表示顶点，第三个数表示权重）

程序结果所得最小生成树以边集的形式（顶点二元组）给出。

## **附录：实验报告**

# 实验题目

学号2019011777

姓名 刘康来

班级 计算机19-3

时间 20、10、16

# 实验题目解答

（对解题的整体思路、过程进行提炼和描述，包括算法描述、程序结构、主要变量说明、设计技巧、调试情况、运行结果、心得体会等）

实验关键是找回路：

按照集合，连通分支的思想，定义一数组a【n+1】（因n个节点会有a【n】出现，需n+1），先全赋为0，表示只在自己的集合中，后有v1和v2，只要v1，v2不在同一集合中就可作为树的边，进而和在同一集合中

- a【n】若=0，则可加边（但要注意环）

- a【v1】与a【v2】分四种情况讨论：0 非0，0 0，非0 0 ，非0 非0

- 0与0需要将v1与v2分在同一集合中，方法：将a【v1】与a【v2】赋同一值

- 0与非0，只需将非0值赋0即可，代表一个jied只在自己集合中，另一个已与其他节点在一集合中

- 非0与非0，代表两节点在不同集合中，需合并集合，可用循环将一个集合中全部值赋为另一非0值。

错误：循环赋值合并集合时，需记录待赋节点值，否则赋第一个值后，后面无法赋值。

# 附：源程序

#include<stdio.h>

int n,m;

int main(void)

{

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

struct Edge{

int v1,v2,weight;

}ex[m];

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

scanf("%d %d %d",&ex[i].v1,&ex[i].v2,&ex[i].weight);

}

// sort the ex by weight

// the node that isn't in tree is 0

// the length !

int a[n+1];

for(int i = 0;i < n+1;i++){

a[i] = 0;

}

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

if(ex[i].v1 ==ex[i].v2) continue; // delete ring

if(a[ex[i].v1] && a[ex[i].v2] && a[ex[i].v1] == a[ex[i].v2]) continue;

if(!a[ex[i].v1]){

if(!a[ex[i].v2]) a[ex[i].v2] = a[ex[i].v1] = i+1;

else a[ex[i].v1] = a[ex[i].v2];

}

else{

if(!a[ex[i].v2]) a[ex[i].v2] = a[ex[i].v1];

else{

int s = a[ex[i].v2];//records the value...

for(int j = 0;j < n+1;j++){

if(a[j] == s) a[j] = a[ex[i].v1];

}

}

}

//printf("%d %d\n",a[ex[i].v1],a[ex[i].v2]);

printf("%d %d\n",ex[i].v1,ex[i].v2);

}

return 0;

}

1 4 1 2 4 1 5 7 1 1 6 2 2 3 2 2 5 2 4 7 2 6 7 2 7 8 2 1 2 3 2 7 3 3 8 3 3 5 4

