电子科技大学信息与软件工程学院

实 验 报 告

学　　号 2016220304022

姓　　名 罗悦

（实验）　课程名称 数据结构与算法

理论教师 陈安龙

实验教师 陈安龙

电子科技大学教务处制表

电 子 科 技 大 学

实　验　报　告（3）

学生姓名：罗悦　　**学　号：**2016220304022　　**指导教师：**陈安龙

实验地点：清水河科技实验大楼　　**实验时间：**2017.5.20

一、实验室名称：学校实验中心软件实验室

二、实验项目名称：编程实现最小生成树 Kruskal 算法

三、实验学时：4

四、实验原理：

Kruskal 算法是一种按照图中边的权值递增的顺序构造最小生成树的方法。其基本思想是：设无向连通网为 G＝（V，E），令 G 的最小生成树为 T，其初态为 T＝（V，{}），即开始时，最小生成树 T 由图 G 中的 n 个顶点构成，顶点之间没有一条边，这样 T 中各顶点各自构成一个连通分量。然后，按照边的权值由小到大的顺序，考察 G 的边集 E 中的各条边。若被考察的边的两个顶点属于 T 的两个不同的连通分量，则将此边作为最小生成树的边加入到 T 中，同时把两个连通分量连接为一个连通分量；若被考察边的两个顶点属于同一个连通分量，则舍去此边，以免造成回路，如此下去，当 T 中的连通分量个数为 1 时，此连通分量便为 G 的一棵最小生成树。

如教材 153 页的图 4.21(a)所示，按照 Kruskal 方法构造最小生成树的过程如图 4.21 所示。在构造过程中，按照网中边的权值由小到大的顺序，不断选取当前未被选取的边集中权值最小的边。依据生成树的概念，n 个结点的生成树，有 n－1 条边，故反复上述过程，直到选取了 n－1 条边为止，就构成了一棵最小生成树。

五、实验目的：

本实验通过实现最小生成树的算法，使学生理解图的数据结构存储表示，并能理解最小生成树 Kruskal 算法。通过练习，加强对算法的理解，提高编

程能力。

六、实验内容：

（1）假定每对顶点表示图的一条边，每条边对应一个权值；

（2）输入每条边的顶点和权值；

（3）输入每条边后，计算出最小生成树；

（4）打印最小生成树边的顶点及权值。

七、实验器材（设备、元器件）：

PC 机一台，装有 C 语言集成开发环境。

八、数据结构与程序：

```
#include<stdio.h>
#define MaxVertexNum 100     //最大顶点数


typedef char VertexType;     //顶点类型
typedef int EdgeType;        //边上的权值

typedef struct{
    VertexType vexs[MaxVertexNum];                  //顶点表
    EdgeType edges[MaxVertexNum][MaxVertexNum];   //邻接矩阵
    int n,e;                                       //图中当前的顶点数和边数
}AdjGragh;

typedef struct{
    int u;                 //边的起始顶点
    int v;                 //边的终止顶点
    int w;                 //边的权值
}Edge;

AdjGragh G;
int n;
Edge E[MaxVertexNum];


void CreateMGraph(){                                    //创建图
    int i,j,k,w,m;
    printf("请输入图中的顶点数：\n");
    scanf("%d",&G.n);
    n=G.n;
```

```
    printf("请输入图中的边数：\n");getchar();
    scanf("%d",&G.e);
    printf("请依次录入顶点:\n");getchar();
    for(i=0;i<G.n;i++){                                  //建立顶点表
        G.vexs[i]=getchar();getchar();

    }
    for(i=0;i<G.n;i++){
            for(j=0;j<G.n;i++){
            G.edges[i][j]=0;                            //邻接矩阵初始化
        }

    }
    printf("请按升序依次输入每条边上的权值 w：\n");
    printf("i  j  w\n");
    m=0;
    for(k=0;k<G.e;k++){    //输入边（vi,vj)上的权 w

            scanf("%d  %d  %d",&E[m].u,&E[m].v,&E[m].w);
            m=m+1;
    }

}

void up(){
    int i,j;
    for(i=0;i<=n-2;i++){
        for(j=i;j<=n-2;j++){
            if(E[i].w>E[j].w){
                E[i]=E[j];
            }
        }
    }
}

void MiniSpanTree_Kruskal(){
    int i,j,k,a;
    int vset[MaxVertexNum];
    for(i=0;i<n;i++){
        vset[i]=i;
    }
    k=1;j=0;
    while(k<n){                                         //生成的边数小于 n 时循环
         if(vset[E[j].u]!=vset[E[j].v]){
```

```
            printf("(%d,%d):%d\n",E[j].u,E[j].v,E[j].w);
            k++;
            for(i=0;i<n;i++){                    //两个集合统一编号
                  if(vset[i]==E[j].v){
                   vset[i]=vset[E[j].u];
                  }
        }
        for(i=0;i<n;i++){
        printf("%d\t",i);
        }
        printf("\n");
        for(i=0;i<n;i++){
        printf("%d\t",vset[i]);
        }
        printf("\n");
        }
        j++;                                     //扫描下一条边
    }

}



int main(){
    CreateMGraph();
    MiniSpanTree_Kruskal();
    return 0;

}
```

九、程序运行结果：

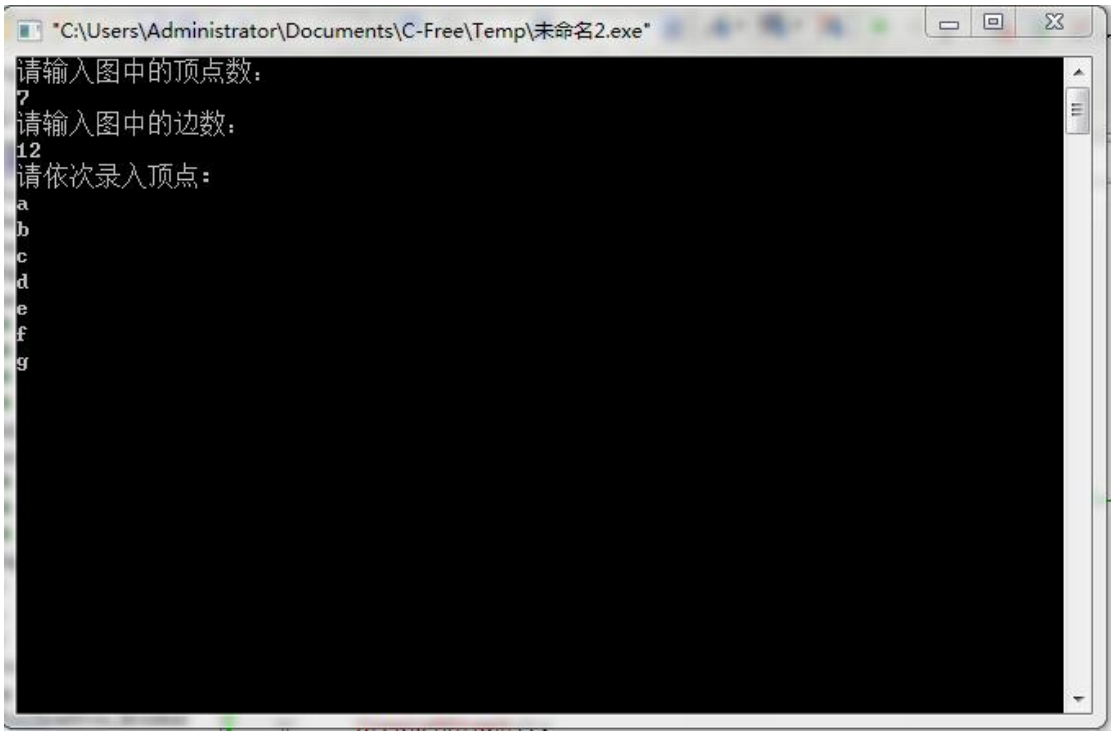

图 1-1

图 1-1 先录入实验图的顶点数、边数，并依次录入顶点。

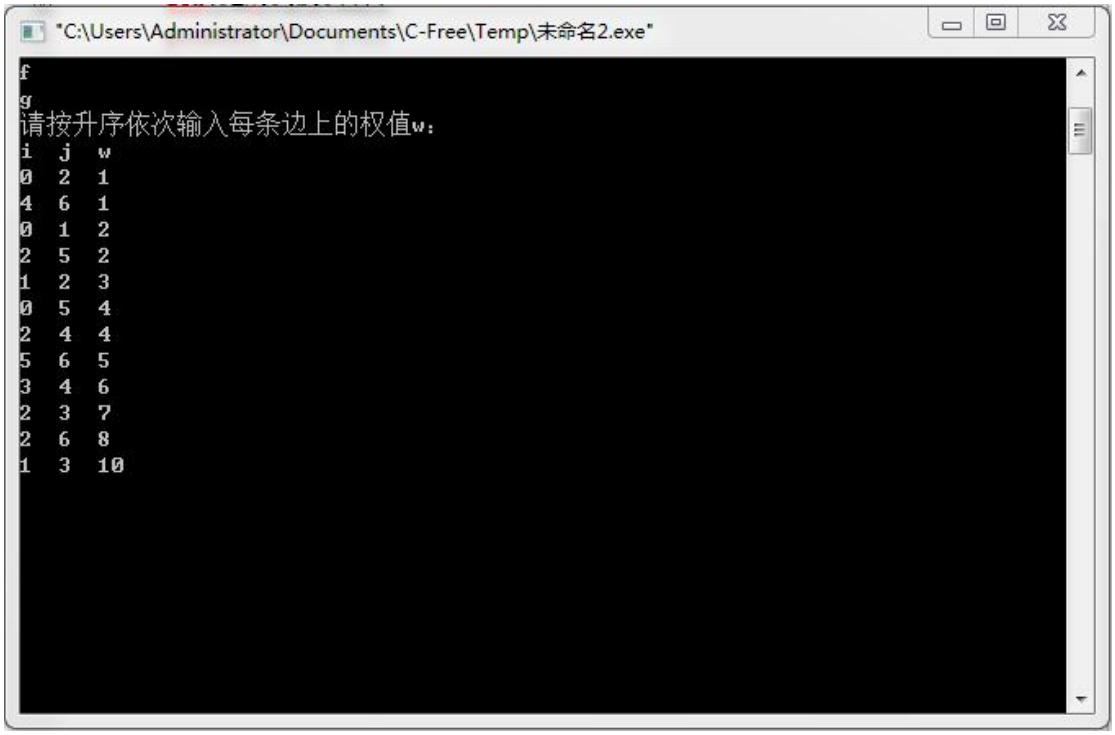

图 1-2

图 1-2 按权值升序依次录入每条边的顶点和权重值。

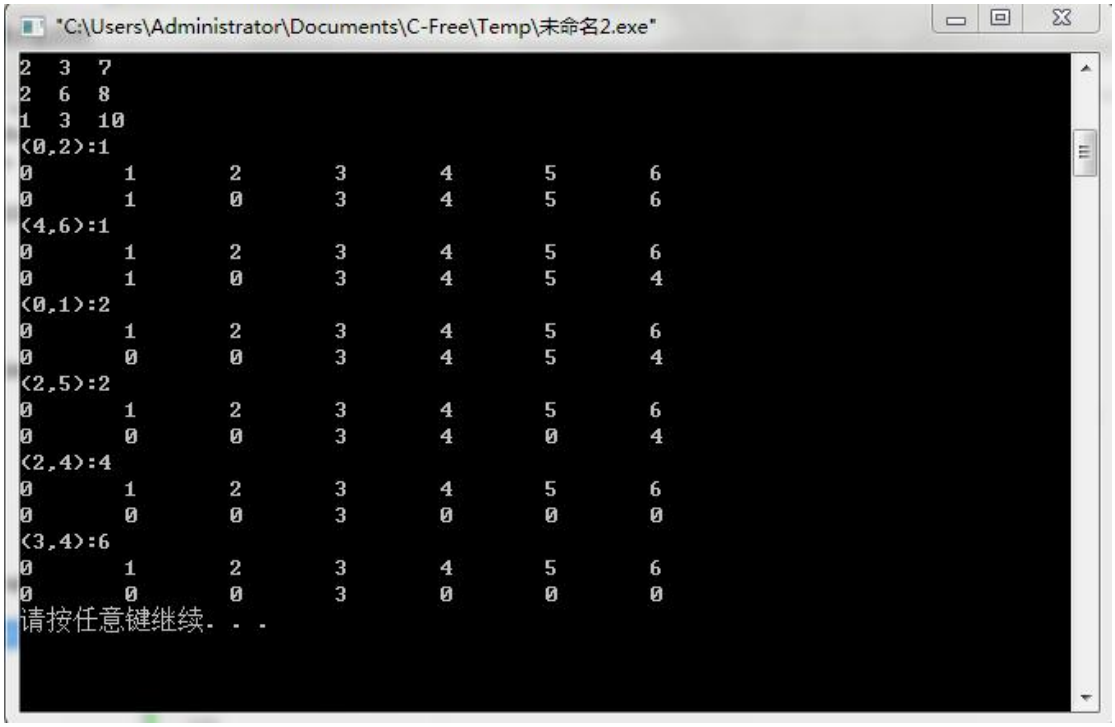

图 1-3

图 1-3 表示每一步所连接的边的顶点和它的权值。即最终所连接的边和它们的权值为：

⟨0, 2⟩	⟨4, 6⟩	⟨0, 1⟩	⟨2, 5⟩	⟨2, 4⟩	⟨3, 4⟩
1	1	2	2	4	6

十、实验结论：

此实验运用 Kruskal 算法从最小的边入手，并从权重最小的边依次找上去，一旦有循环便去掉此条边并跳到下一条边。并最终得到一个最小生成树，使得实验成功。

十一、总结及心得体会：

Kruskal 算法按照图中边的权值递增的顺序构造最小生成树，但在选择边的过程中会遇到会形成回路，此时便会考察边的两个顶点是否属于同一个连通分量，如果属于同一个连通分量，则舍去此条边，最终便会得到一个没有回路的最小生成树。本次试验使我理解了图的数据结构存储表示，并能理解最小生成树 Kruskal 算法。并且加强了对算法的理解，提高了编程能力。在编程的过程中应当理清楚每一步，每一个循环的作用，这次试验的循环语句较多，容易出现错误，应当小心看好每一步，谨慎的完成实验。Kruskal 算法与 Prim 算法正好相反，但两个算法都运用了集合的方法，得到最小生成树。