**实验报告（第 7 次）**

**实验名称：最小生成树两种算法的设计与实现**

**实验时间： 2021.5.21**

1. **实验目的**

1．掌握最小生成树的定义和构造方法；

2．掌握最小生成树的两种常用生成算法及效率分析。

1. **实验内容**

编程实现：

1、建立一个以邻接矩阵形式存储的图；

2、以Prim算法获得最小生成树；

3、以Kruscal算法获得最小生成树；

4、退出系统。

回答下述问题：

1、分析两种算法的时间效率和空间效率；

2、有向网如何实现求最小生成树？

1. **源程序及主要算法说明**

**/\*\***

**\*202001010193 刘欢**

**\*/**

**#include<stdio.h>**

**#define MAX 100**

**#define MAXCOST 0x7fffffff**

**int graph[MAX][MAX];**

**void prim(int graph[][MAX],int n)**

**{**

**int lowcost[MAX];**

**int mst[MAX];**

**int i, j,min,minid,sum=0;**

**for(i=2;i<=n;i++)**

**{**

**lowcost[i]=graph[1][i];**

**mst[i];**

**}**

**mst[1]=0;**

**for(i=2;i<=n;i++)**

**{**

**min=MAXCOST;**

**minid=0;**

**for(j=2;j<=n;j++)**

**{**

**if(lowcost[j]<min&&lowcost[j]!=0)**

**{**

**min=lowcost[j];**

**minid=j;**

**}**

**}**

**printf("v%d-v%d\n",mst[minid],minid,min);**

**sum+=min;**

**lowcost[minid]=0;**

**for(j=2;j<=n;j++)**

**{**

**if(graph[minid][j]<lowcost[j])**

**{**

**lowcost[j]=graph[minid][j];**

**mst[j]=minid;**

**}**

**}**

**}**

**printf("最小权值之和=%d\n",sum);**

**}**

**int main()**

**{**

**int i, j, k, m, n;**

**int x, y, cost;**

**printf("请输入顶点个数，边的个数：");**

**scanf("%d%d",&m,&n);//m=顶点的个数，n=边的个数**

**for (i = 1; i <= m; i++)//初始化图**

**{**

**for (j = 1; j <= m; j++)**

**{**

**graph[i][j] = MAXCOST;**

**}**

**}**

**for (k = 1; k <= n; k++)**

**{**

**printf("请输入顶点号，权值：");**

**scanf("%d%d%d",&i,&j,&cost);**

**graph[i][j] = cost;**

**graph[j][i] = cost;**

**}**

**prim(graph, m);**

**return 0;**

**}**

**/\*\***

**\*202001010193 刘欢**

**\*/**

**#include <stdio.h>**

**#define MAXE 100**

**#define MAXV 100**

**typedef struct{**

**int vex1; //边的起始顶点**

**int vex2; //边的终止顶点**

**int weight; //边的权值**

**}Edge;**

**void kruskal(Edge E[],int n,int e)**

**{**

**int i,j,m1,m2,sn1,sn2,k,sum=0;**

**int vset[n+1];**

**for(i=1;i<=n;i++) //初始化辅助数组**

**vset[i]=i;**

**k=1;//表示当前构造最小生成树的第k条边，初值为1**

**j=0;//E中边的下标，初值为0**

**while(k<e)//生成的边数小于e时继续循环**

**{**

**m1=E[j].vex1;**

**m2=E[j].vex2;//取一条边的两个邻接点**

**sn1=vset[m1];**

**sn2=vset[m2];**

**//分别得到两个顶点所属的集合编号**

**if(sn1!=sn2)//两顶点分属于不同的集合，该边是最小生成树的一条边**

**{//防止出现闭合回路**

**printf("V%d-V%d=%d\n",m1,m2,E[j].weight);**

**sum+=E[j].weight;**

**k++; //生成边数增加**

**if(k>=n)**

**break;**

**for(i=1;i<=n;i++) //两个集合统一编号**

**if (vset[i]==sn2) //集合编号为sn2的改为sn1**

**vset[i]=sn1;**

**}**

**j++; //扫描下一条边**

**}**

**printf("最小权值之和=%d\n",sum);**

**}**

**int fun(Edge arr[],int low,int high)**

**{**

**int key;**

**Edge lowx;**

**lowx=arr[low];**

**key=arr[low].weight;**

**while(low<high)**

**{**

**while(low<high && arr[high].weight>=key)**

**high--;**

**if(low<high)**

**arr[low++]=arr[high];**

**while(low<high && arr[low].weight<=key)**

**low++;**

**if(low<high)**

**arr[high--]=arr[low];**

**}**

**arr[low]=lowx;**

**return low;**

**}**

**void quick\_sort(Edge arr[],int start,int end)**

**{**

**int pos;**

**if(start<end)**

**{**

**pos=fun(arr,start,end);**

**quick\_sort(arr,start,pos-1);**

**quick\_sort(arr,pos+1,end);**

**}**

**}**

**int main()**

**{**

**Edge E[MAXE];**

**int nume,numn;**

**printf("输入顶数和边数:\n");**

**scanf("%d%d",&numn,&nume);**

**for(int i=0;i<nume;i++)**

**{**

**printf("请输入顶点号，权值：");**

**scanf("%d%d%d",&E[i].vex1,&E[i].vex2,&E[i].weight);**

**}**

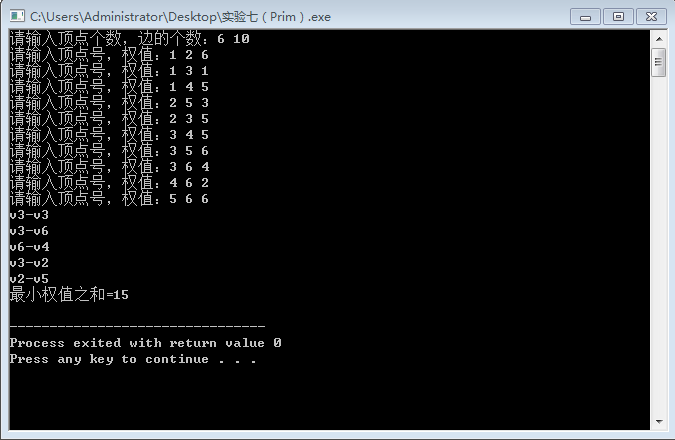
**quick\_sort(E,0,nume-1);**

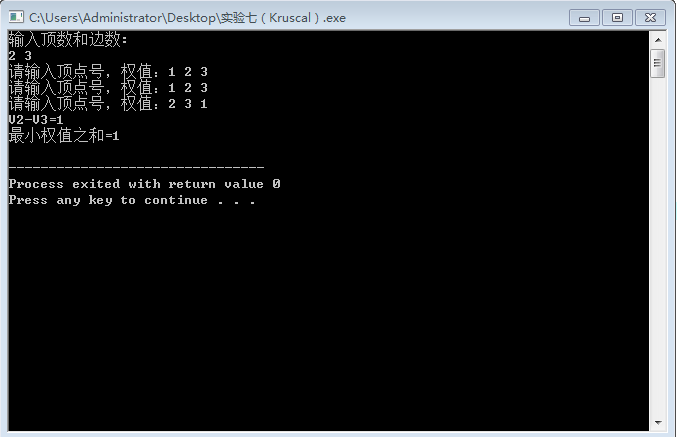
**kruskal(E,numn,nume);**

**}**

**四、主要问题和解决方案**

1. **测试数据及结果**

****

****

1. **心得体会与自我评价**

这次试验挺简单的，比较容易懂，有进步。

1. **教师评分**