数据结构实验报告（三）

班级：计科1194 姓名：万一凡 学号：201911111420

实验日期：2020年11月25日

实验名称：图的基本操作和应用

实验内容：1.邻接矩阵、邻接表两种方式创建图  
 2.深度遍历、广度遍历  
 3.最小生成树

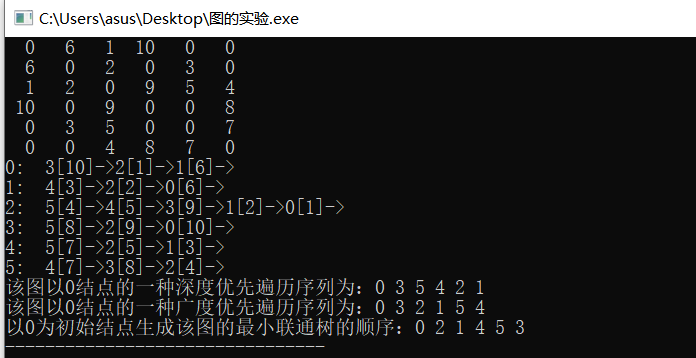
算法描述或算法思路：

1. 运用邻接矩阵组表示顶点之间相邻关系，给定一个邻接矩阵组，和定点数和边数就可以直接生成了。邻接表是先创建一个包含所有顶点信息的数组，然后每个数组里的顶点建立一个单链表，就是将顶点所有邻接点连起来，每个边结点与对应的头结点之间表示一条边的信息。
2. 深度优先遍历：通俗的说先一条路走到底，然后返回岔路口再走另一条路。过程类似与树的前序遍历，先访问初始点，然后判断初始点的邻接点有没有被访问过，如果没有，递归访问该邻接点，然后再循环判断初始点的其他邻接点。

广度优先遍历：先访问初始点，然后访问初始点的所有邻接点，再访问所有邻接点的全部邻接点，类似于树的层序遍历。需要用到队列，首先初始点进队，然后初始点出队的时候，它的邻接点进队。如此反复知道队列为空。即当有顶点出队的，它的邻接点就要进队，所有点进队之后都要按顺序出队。

3.最小生成树：采用Prim算法，首先将初始点当成一个集合，其他点为一个集合。找到这两个集合间的最小路径(带权边)，将最小带权边的顶点加入初始点的集合。如此反复寻找两个集合之间的最小路径，不停的往初始点的集合里加入顶点，直到所有顶点都加入了进去。

运行情况及分析:



程序运行结果达到预期，实验采用无向带权图，邻接矩阵里0代表没有边，其他数代表权值。只是深度优先序列和广度优先序列，只能给出一种。

源代码：

**#include <stdio.h>**

**#include <stdlib.h>**

**#define max 100**

**struct bian{**//邻接表里边界点的结构

**int data;**

**struct bian \*next;**

**int weight;**

**};**

**struct tou{**//头结点结构

**bian \*nextb;**//指向边结点的指针

**int data;**//头结点的值

**};**

**struct excel{**//邻接表类型

**tou data[max];**//头结点的数组

**int n,e;**

**};**

**struct matrix**{//邻接矩阵,n为结点数，e为边数

**int num[max][max];**

**int n,e;**

**};**

**void create(matrix &p,int a[max][max],int n,int e){**

**int i,j;**

**p.n=n;**//存储结点数和边数

**p.e=e;**

**for(i=0;i<p.n;i++){**//复制数组

**for(j=0;j<p.n;j++){**

**p.num[i][j]=a[i][j];**

**}**

**}**

**}**

**void dis(matrix p){**//打印邻接矩阵

**int i,j;**

**for(i=0;i<p.n;i++){**

**for(j=0;j<p.n;j++){**

**printf("%3d ",p.num[i][j]);**

**}**

**printf("\n");**

**}**

**}**

**void createexcel(excel \*&p,int a[max][max],int n,int e){**//创建邻接表

**int i,j;** //本实验采用无向带权图图

**p=(excel\*)malloc(sizeof(excel));**

**bian \*r;**

**for (i=0;i<n;i++){**  // 先将头结点的数组初始化

**p->data[i].nextb=NULL;** //将后续指针指向空，

**}**

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

**for(j=0;j<n;j++){**

**if(a[i][j]>0){**//判断边是否存在

**r=(bian\*)malloc(sizeof(bian));**

**r->data=j;**

**r->weight=a[i][j];**

**r->next=p->data[i].nextb;**//采用头插法插入边的信息

**p->data[i].nextb=r;**

**}**

**}**

**}**

**p->n=n;p->e=e;**

**}**

**void display(excel\*p){**//输出邻接表

**bian \*r;**

**r=(bian\*)malloc(sizeof(bian));**

**int i;**

**for (i=0;i<p->n;i++){**

/\*先从头结点进去，输出边结点\*/

**r=p->data[i].nextb;**

**printf("%d: ",i);**

**while (r!=NULL){**

**printf("%d[%d]->",r->data,r->weight);**//输出值和权重，中括号内为权

**r=r->next;**

**}**

**printf("\n");**

**}**

**}**

**void dfs(excel \*p,int n,int visit[max]){**//深度优先遍历

**bian \*r;**

**visit[n]=1;**//访问置标记

**printf("%d ",n);**

**r=p->data[n].nextb;**//指向该值头顶点的第一个边结点

**while(r!=NULL){**

**if(visit[r->data]==0){**//如果没有访问过，递归访问边结点

**dfs(p,r->data,visit);**

**}**

**r=r->next;**//指向下个一边结点

**}**

**}**

**void bfs(excel \*p,int n,int visit[max]){**//广度优先遍历

**int i;**

**for(i=1;i<max;i++){**//初始化访问数组

**visit[i]={0};**

**}**

**int rank[max]={-1};**//定义队列

**int head,tail;**//head为头，tail为尾

**head=tail=0;**

**rank[tail]=n;**

**tail++;**

**printf("%d ",rank[head]);**//先打印首结点

**visit[n]=1;**//标志已访问

**bian \*r;**

**r=(bian\*)malloc(sizeof(bian));**

**do{**

**r=p->data[rank[head]].nextb;**

**head++;**//抽象出队，实际并没有

**while(r!=NULL){**

**if(visit[r->data]==0){**//如果没有访问

**visit[r->data]=1;**//标记

**printf("%d ",r->data);**//访问

**rank[tail]=r->data;**//进队

**tail++;**

**}**

**r=r->next;**//指向下一个边结点

**}**

**}while(tail>head);**//队列不为空时循环；

**printf("\n");**

**}**

**void prim (excel \*p,int n){**//最小生成树的prim算法

**int visit[max]={0};**

**int result[p->n]={-1};**//用来存放生成的结果

**int count=0;**//形成边的计数

**result[count]=n;**//将初始点放入

**visit[n]=1;//标志**

**bian \*r,\*min;//min**用来标志两集合间最小权重的边结点。

**min=(bian\*)malloc(sizeof(bian));**//初始化

**min->next=NULL;**

**int i=0;**

**while(count<p->n){**//n个结点要生成n-1条边，所以以此控制

**min->weight=100000;**//给一个稍大的权值报证图的权值没有超过辞职的

**for(i=0;i<=count;i++){**//下两个循环是寻找两个集合间的最小权重的过程

**r=p->data[result[i]].nextb;**//所有结果的头结点开始

**while(r!=NULL){**//判断边结点的权重 ，寻找最小值

**if(min->weight>r->weight&&visit[r->data]==0){**

**min=r;**

**}**

**r=r->next;**

**}**

**}**

**visit[min->data]=1;**//找到最小值后，生成之后进行标记

**count++;**//生成边数加1

**result[count]=min->data;**//存放生成的结果

**}**

**for(i=0;i<count;i++){**

**printf("%d ",result[i]);**

**}**

**}**

**int main (){**

**int a[max][max]={**//带权无向图

**{0,6,1,10,0,0},{6,0,2,0,3,0},**

**{1,2,0,9,5,4},{10,0,9,0,0,8},**

**{0,3,5,0,0,7},{0,0,4,8,7,0}};**

**int n=6,e=10;**

**excel \*p;**

**matrix s;**

**create(s,a,n,e);**

**dis(s);**

**createexcel(p,a,n,e);**

**display(p);**

**int visit[max]={0};**

**printf("该图以0结点的一种深度优先遍历序列为：");**

**dfs(p,0,visit);**

**printf("\n");**

**printf("该图以0结点的一种广度优先遍历序列为：");**

**bfs(p,0,visit);**

**int i=0;**

**printf("以0为初始结点生成该图的最小联通树的顺序：");**

**prim(p,i);**

**return 1;**

**}**

实验体会：

这些实验要将理论转化为程序实践，图是较为复杂的数据结构，所有在实践时遇到一点困难，但经过反复在纸上的推算和程序调试，初步完成一些基本功能，其中有点地方实略有不足。

其中深度遍历和广度遍历还利用之前学习过的二叉树的思想才得以实现。邻接表的建立也使用了单链表的结构，所有要一步步的掌握并熟练使用。