中北大学软件学院

**实验报告**

|  |  |
| --- | --- |
| 专 业： | 软件工程 |
| 课程名称： | 数据结构与算法 |
| 班 级： | 21130420 |
| 学 号： | 2113042015 |
| 姓 名： | 程楚晋 |
| 辅导教师： | 马巧梅 |

2022年03月制

成绩：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 实验时间 | 2022年5月25日7 时至 9时 | 学时数 | 2学时 | |
| 1.实验名称  **图的应用** | | | | |
| 2.实验目的  （1）掌握图的结构特性，各种存储结构及其适用范围；  （2）掌握图的创建及遍历算法。  （3）能够使用图解决实际问题，计算最小代价生成树。 | | | | |
| 3.实验内容  **基本内容：**  算法1：输入图的类型、顶点数、狐（边）数、顶点信息、狐（边）信息，建立相应的图(具体类型可以是无向图、有向图、无向网、有向网，采用邻接矩阵存储结构)；分别按深度优先搜索和广度优先搜索遍历图；按某种形式输出图及遍历结果。  **选作内容：**  算法2：采用邻接表存储结构，其他要求同算法1。  算法3：完成计算图的最小代价生成树的算法。 | | | | |
| 4.拟遍历的图的初态及存储结构（此处画出你计划遍历的图及所采用的存储结构）  遍历的图： 该图的矩阵存储形式 | | | | |
| 5.实验源代码  #include<stdio.h>  #include "stdlib.h"  #include<malloc.h>  #define OK 1  #define ERROR 0  #define OVERFLOW -1  #define TRUE 1  #define FALSE 0  #define MVNum 100  #define MAXSIZE 100  #define INFINITY 32767 //表示无穷大  typedef struct {  char vertex[MVNum]; //顶点信息  int arcs[MVNum][MVNum]; //边信息  int vex\_num,arc\_num; //图当前的点数和边数  }Mgraph;  struct Prim{  char adjvex;  int lowcoat;  }closedge[20];  typedef struct{  int \*data;  int front;  int rear;  }SqQueue;  void InitQueue(SqQueue &Q)  {  Q.data=(int \*)malloc(sizeof(int)\*MAXSIZE);  if(!Q.data) exit(-2);  Q.front=Q.rear=0;  }  int QueueEmpty(SqQueue Q)  {  if(Q.front==Q.rear)  return 1;  else  return 0;  }  int EnQueue(SqQueue &Q,int e)  {  if((Q.rear+1)%MAXSIZE==Q.front)  return 0;  Q.data[Q.rear]=e;  Q.rear=(Q.rear+1)%MAXSIZE;  return 1;  }  int DeQueue(SqQueue &Q,int &e)  {  if (Q.front==Q.rear)  return 0;  e=Q.data[Q.front];  Q.front=(Q.front+1)%MAXSIZE;  }  //查找位置  int LocateVex(Mgraph g,char v)  {  int i;  for(i=0; i<g.vex\_num; ++i)  {  if(g.vertex[i]==v)  return i;  }  return -1;  }  //图的创建  void creat\_UDG(Mgraph &g)  {  int i,j,k;  char v1,v2;  int cost;  printf("请输入顶点个数，边的个数：\n");  scanf("%d,%d",&g.vex\_num,&g.arc\_num);  getchar(); //吃掉垃圾字符  printf("请输入顶点信息：\n");  //构造顶点向量  for(i = 0; i< g.vex\_num ; ++i)  {  scanf("%c",&g.vertex[i]);  getchar();  }  //初始化邻接矩阵  for(i=0; i<g.vex\_num; ++i)  for(j=0; j<g.vex\_num;++j)  g.arcs[i][j] = INFINITY;  printf("请输入边的信息：\n");  //构造邻接矩阵  for(k=0; k<g.arc\_num; ++k)  {  //输入一条边依附的顶点  scanf("%c,%c,%d",&v1,&v2,&cost);  getchar();  //确定v1和v2在G中的位置  i = LocateVex(g,v1);  j = LocateVex(g,v2);  if(i == -1 || j == -1) return;  g.arcs[i][j] = cost;  g.arcs[j][i] = g.arcs[i][j];  }  }  //图的深度搜索遍历  bool vistit[MVNum];  char DFS(Mgraph G,int v)//这里的v是输入结点的序号  {  printf("%d ",v);  vistit[v] = true;  int w;  printf("%c ",G.vertex[v]) ;  for(w=0;w<G.vex\_num;w++)  if((G.arcs[v][w] != 0)&&(!vistit[w])&&G.arcs[v][w] != 32767) DFS(G,w);  }  //图的广度搜索遍历  void BFS(Mgraph G)  {  SqQueue Q;  char e;  InitQueue(Q);  for(int i = 0; i < G.vex\_num;i++)  {  vistit[i] = false;  }  for(int i = 0;i<G.vex\_num;i++) {  if (!vistit[i]) {  vistit[i] = true;  printf("%c ",G.vertex[i]);  EnQueue(Q, e);  while(!QueueEmpty(Q))  {  DeQueue(Q,i);  for(int j = 0;j<G.vex\_num;j++) {  if (G.arcs[i][j] != INFINITY && !vistit[j]) {  vistit[j] = true;  printf("%c ", G.vertex[j]);  EnQueue(Q, j);  }  }  }  }  }  }  //图的显示算法函数  void out\_UDG(Mgraph g)  {  int i,j;  printf("图的顶点是：\n");  for(i=0; i<g.vex\_num;i++)  {  printf("%8c",g.vertex[i]);  }  printf("\n");  printf("图的邻接矩阵是：\n");  for(i=0; i<g.vex\_num; i++)  {  for(j=0; j<g.vex\_num; j++)  /\*if(g.arcs[i][j]==32767) printf("\t∞");  else printf("\t%d", g.arcs[i][j]);  printf("\n");\*/  printf("%8d",g.arcs[i][j]);  printf("\n");  }  }  int main()  {  Mgraph g;  int x;  creat\_UDG(g);  out\_UDG(g);  printf("图的深度优先遍历\n");  printf("输入从某节点遍历的序号");  puts("");  scanf("%d",&x);  DFS(g,x);  puts("");  printf("广度优先遍历");  puts("");  BFS(g);  return 0;  }  //运行样例 | | | |
| 6.实验结论及心得  在这次实验中，我发现一个程序的实现并不能拘泥于课本上的或者老师讲的方法上面，若想完整的写出一个程序，不光要将前面学习的数据结构中的内容，诸如，栈，队列，循环队列等等内容学扎实，还要理解每个，不同数据结构算法的本质知识，有许多方法，需要灵活运用书上的知识，才能将一个程序写好，  在程序的过程中，首先要理解算法的理论依据，然后观察算法执行过程中d额状态变化，最好是画出每一步的执行步骤，数据直接按的转化关系等等，将这些问题弄清楚后，写的会更加顺手 | | | |