实验项目名称： 图及图的应用

**（所属课程：** 数据结构 **）**

**学 院： 计算机科学与信息工程学院 专业班级：网络工程18-1**

**姓 名： 邓二浦 学 号：18031110129**

**实验日期： 2019-10-11 实验地点：a06-203**

**合作者： 指导教师：孙高飞**

**本实验项目成绩： 教师签字： 日期：**

1. **实验目的**

理解图的基本概念，两种主要的存储结构。掌握在邻接链表存储结构下的图的深度优先递归遍历、广度优先遍历。通过选做题"最短路径问题"认识图及其算法具有广泛的应用意义。

1. **实验内容**

2.1 在邻接矩阵存储结构下的图的深度优先递归遍历、广度优先遍历。

2.2 拓扑排序，求图的拓扑序列

**三 、实验步骤与结果**

2.1 实验步骤

* 要完成图的两种遍历算法，首先需要进行图的数据初始化。为把时间主要花在遍历算法的实现上，图的初始化采用结构体声明时初始化的方法。示例代码如下：

#include "stdio.h"

typedef int Arcell;

typedef int AdjMatrix[5][5];

typedef struct {

char vexs[5];

AdjMatrix arcs;

int vexnum,arcnum;

}MGraph;

void main(){

MGraph g={

{'a','b','c','d','e'},

{{0,1,0,1,0},

{1,0,0,0,1},

{1,0,0,1,0},

{0,1,0,0,1},

{1,0,0,0,0}} ,5,9};

}

* 录入调试算法7.5 深度优先遍历图，输出顶点遍历顺序。

#include "stdio.h"

#define max 100

typedef int Arcell;

typedef int AdjMatrix[5][5];

typedef struct {

char vexs[5];

AdjMatrix arcs;

int vexnum,arcnum;

}MGraph;

void dfs(MGraph g,int v,int visited[]) //递归深度优先搜索遍历

{

int j;

printf("%c ",g.vexs[v]);

visited[v]=1;

for(j=0;j<g.vexnum;j++)

{

if(!visited[j]&&g.arcs[v][j]==1)

dfs(g,j,visited);

}

}

void main(){

int v,visited[max],i;

MGraph g={

{'a','b','c','d','e'},

{{0,1,0,1,0},

{1,0,0,0,1},

{1,0,0,1,0},

{0,1,0,0,1},

{1,0,0,0,0}},5,9};

printf("请输入从哪个顶点开始遍历，0-4分别代表a和e\n");

scanf("%d",&v);

for(i=0;i<g.vexnum;i++)

visited[i]=0; //初始时，全部没有被访问过

dfs(g,v,visited);

for(i=0;i<g.vexnum;i++)//图中可能存在连通子图

if(visited[i]==0)

dfs(g,i,visited);

}

* 录入调试算法7.8广度优先遍历图， 输出顶点遍历顺序。

void bfs(MGraph g,int v,int visited[]) //调用一次只能遍历出连通子图中点，如果需要输出整个连通图，需要多次调用，调用几次就有几个连通子图

{

int queue[max],a[max];

int front=0,rear=1,i,j,k;

printf("%c ",g.vexs[v]);

visited[v]=1; //表示已经被访问过

queue[rear]=v;

while(front!=rear)

{

front=(front+1)%max;

v=queue[front];

i=0; //相当于记录出度

for(j=0;j<g.vexnum;j++)

if(g.arcs[v][j]==1)

a[i++]=j;

k=0;

while(k<i)

{

if(visited[a[k]]==0) //如果没有被访问过

{

visited[a[k]]=1; //表示已经访问过

printf("%c ",g.vexs[a[k]]);

rear=(rear+1)%max;

queue[rear]=a[k];

}

k++;

}

}

}

2.2实验步骤

* 录入AdjList 邻接表结构定义
* 录入并调试算法7.11 拓扑排序算法（需要先实现算法7.12求入度算法）

#define max 100

typedef struct node

{

int adjvex;

struct node \*next;

}arcnode;

typedef struct

{

int vertex;

arcnode \*firstarc;

}vexnode;

vexnode adjlist[max];

int creatadjlist()

{

arcnode \*ptr;

int arcnum,vexnum,v1,v2;

printf("\n请输入顶点数和边数\n");

scanf("%d%d",&vexnum,&arcnum);

for(int i=1;i<=vexnum;i++)

{

adjlist[i].firstarc=NULL;

adjlist[i].vertex=0; //各顶点的入度赋初值为0

}

for(i=0;i<arcnum;i++)

{

printf("v1,v2=");

scanf("%d%d",&v1,&v2);

ptr=(arcnode \*)malloc(sizeof(struct node));

ptr->adjvex=v2;

ptr->next=adjlist[v1].firstarc;

adjlist[v1].firstarc=ptr;

adjlist[v2].vertex++;

}

return vexnum;

}

void toposort(int n)

{

int queue[max];

int front=0,rear=0;

int v,w,n1;

arcnode \*p;

n1=0;

for(v=1;v<=n;v++) //循环检测入度为0的顶点并入队

if(adjlist[v].vertex==0)

{

rear++;

queue[rear]=v;

}

printf("拓扑排序的结果：\n");

while(front!=rear)

{

front++;

v=queue[front];

printf("%d ",v);

n1++;

p=adjlist[v].firstarc;

while(p!=NULL)

{

w=p->adjvex;

adjlist[w].vertex--; //将邻接于顶点v的入度减1

if(adjlist[w].vertex==0)

{

rear++;

queue[rear]=w;

}

p=p->next;

}

}

if(n1<n)

printf("构成回路\n");

1. **实验总结**
2. **通过做题中发现，一次广度或者深度遍历，只能遍历图的一个连通子图。通过广度深度算法调用的次数，可以得到一个图中有几个连通子图。**
3. **拓扑排序可以判断图是否是否存在回路。**
4. **邻接矩阵每一行代表他的出度，每一列代表他的入度。**