

《数据结构课程实践》

**实验报告**

**实验名称：图的存储和遍历**

**姓 名：杨天诏**

**学 号：20081831**

**专 业：通信工程**

**实验时间：11月28日**

**杭州电子科技大学**

**通信工程学院**

**（实验报告要求：填写三四五一共三部分内容，实验报告文件命名方式：*数据结构课程实践2022\_实验报告9-姓名-学号*，完成后上传电子版到校园网的网络教学平台，无需打印）**

1. **实验目的**

**掌握构造图的存储，领会图的深度遍历和广度遍历的算法设计。**

1. **实验内容**
2. **（实验指导书186页实验8.1）实现图的矩阵存储和邻接表存储；**
3. **（实验指导书186页实验8.2）实现图的深度优先遍历（只做递归算法）和广度优先遍历。**
4. **（附加题）利用任务2中的代码，用广度优先遍历的方法，找出顶点4到顶点1的路径并输出该路径。**
5. **算法设计**
6. 比较二叉树和图的深度优先遍历算法（二叉树中又称为先序遍历）

先序遍历有左右之分，而深度遍历没有左右之分。而且二叉树通常只用先序、中序、后序。一般图用广度和深度遍历

1. 比较二叉树和图的广度优先遍历算法（二叉树中又称为层次遍历）

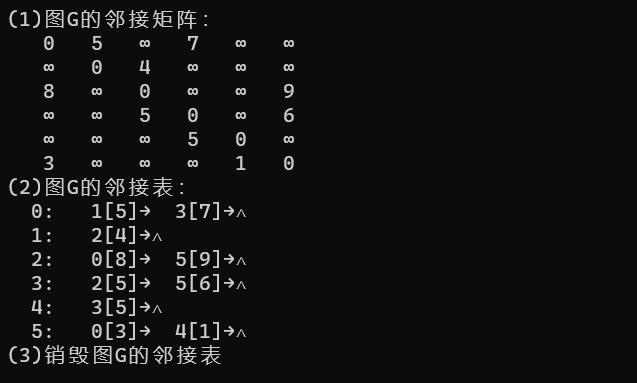
两者都是从一个结点b出发一次访问其相邻结点，对于树来说，就是它的左右孩子结点，而图则是连通的结点。对图来说，一个顶点的相邻结点有多个，而二叉树只有两个。另外，广度遍历图的时候，需要加上一个Visited[MAVX]数组，来记录已访问的结点，避免重复访问同个结点

1. 简述附加题的算法设计思路。

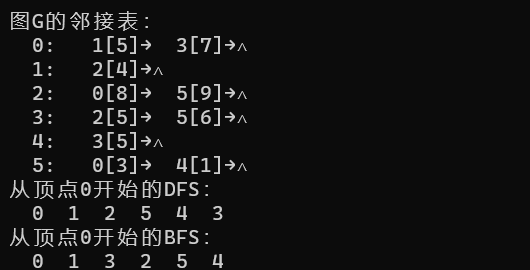
定义一个名为QUERE的结构体，用于存放顶点编号和前一个顶点的位置，用front和rear两个变量作为队头，队尾指针。将起始顶点存入qu，位置记录为-1；出队顶点w，当w与终点相同时，输出路径；不相同时，用广度优先遍历，找到w的所有邻接点并将其分别存入qu中，并标记已遍历；完成后再次出队下一个顶点，直至循环结束。

1. **运行结果与分析（给出运行结果截图，并加以简要说明）**

**题目一：**

****

**题目二：**

****

**附加题：**

**代码：**

#include"y.h"

typedef struct

{

int data;//顶点编号

int parent;//前一个顶点的位置

} QUERE;

void ShortPath(AdjGraph\* G, int u, int v)

{

//输出从顶点u到顶点v的最短路径

ArcNode\* p;

int w, i,x[MAXV],j=0;

QUERE qu[MAXV];//非环形队列

int front = -1, rear = -1;//队列的头，尾指针

int visited[MAXV];

for (i = 0; i < G->n; i++)//访问标记置初值0

{

visited[i] = 0;

}

rear++;

qu[rear].data = u;

qu[rear].parent = -1;

visited[u] = 1;//顶点u进队,表示已经访问过

while (front != rear)//队不空时循环

{

front++;//出队顶点w

w = qu[front].data;

if (w == v)//找到v时输出路径之逆路径

{

i = front;//指向顶点v的位置

while (qu[i].parent != -1)

{

x[j] = qu[i].data;

j++;

i = qu[i].parent;

}

x[j] = qu[i].data;

break;

}

p = G->adjlist[w].firstarc;//找到w的第一个邻接点

while (p != NULL)

{

if (visited[p->adjvex] == 0)

{

visited[p->adjvex] = 1;

rear++; //将w的未访问过的邻接点进队

qu[rear].data = p->adjvex;

qu[rear].parent = front;

}

p = p->nextarc;//找w的下一个邻接点

}

}

for (int t = 0; t<= j; t++)

{

printf("%2d", x[j-t]);

}

}

int main()

{

AdjGraph\* G;

int A[MAXV][MAXV] = {

{0,5,INF,7,INF,INF}, {INF,0,4,INF,INF,INF},

{8,INF,0,INF,INF,9}, {INF,INF,5,0,INF,6},

{INF,INF,INF,5,0,INF}, {3,INF,INF,INF,1,0}

};

int n = 6, e = 10;

CreateAdj(G, A, n, e);

printf("图G的邻接表：\n"); DispAdj(G);

printf("从顶点4开始的BFS：\n");

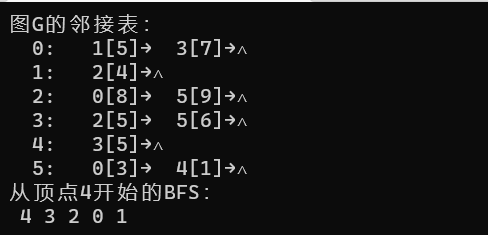
ShortPath(G, 4,1);

printf("\n");

return 0;

}

结果：

****

1. **实验小结（记录实验过程中遇到的主要问题和心得）**

**对图的深度遍历和广度遍历有了更进一步的了解。**