数据结构实验报告

院系：信息工程学院

专业：软件工程二班

学号：20122480212

姓名：霍静静

图的遍历的实验报告

**一、实验目的**：了解图的存储结构（邻接矩阵（数组）存储和邻接表），并可以完成图的遍历；

**二、实验要求**：

1）定义图的存储结构；

2）熟练掌握图的遍历的思想和算法；

**三、实验内容：**

**创建无向图或者有向图，完成图的深度优先遍历或者广度优先遍历；本实验我选择的是创建无向图并完成深度优先遍历；**

**四、算法思想：**

**首先，根据用户输入的顶点总数和边数，构造无向图，然后以用户输入的顶点为起始点，进行深度优先遍历（递归算法），并输出遍历的结果。**

**五、主要函数的实现：**

**#include <stdio.h>**

**#include <string.h>**

**#include <stdlib.h>**

**#include <limits.h>**

**#define MAX\_VERTEX\_NUM 20**

**typedef int VRType;**

**typedef char VertexType;**

**typedef enum {DG,UDG} GKind; //{有向图,无向图}**

**typedef struct**

**{**

**VertexType vexs[MAX\_VERTEX\_NUM]; //顶点向量**

**VRType arcs[MAX\_VERTEX\_NUM][MAX\_VERTEX\_NUM];//邻接矩阵**

**int vexnum,arcnum; //顶点数和弧数**

**GKind kind; //图的种类标识**

**}MGraph;**

**int \*visited; //记录顶点是否被访问**

**//返回字符在定点向量中的下标值**

**int LocateVex(const MGraph &G,VertexType ch)**

**{**

**int i;**

**for ( i = 0; G.vexnum; i++ )**

**if ( ch == G.vexs[i] )**

**return i;**

**return -1;**

**}**

**//得到第一个未被访问的相邻节点下标，若无，则返回-1**

**int firstAjdVex(const MGraph &G,int v)**

**{**

**int i;**

**for ( i = 0; i < G.vexnum; i++ )**

**{**

**if ( !visited[i] && G.arcs[v][i] > 0 && G.arcs[v][i] < INT\_MAX)**

**return i;**

**}**

**return -1;**

**}**

**//得到下一个未被访问的相邻节点下标，若无，则返回-1**

**int nextAjdVex(const MGraph &G,int v,int w)**

**{**

**int i;**

**for ( i = w; i < G.vexnum; i++ )**

**{**

**if ( !visited[i] && G.arcs[v][i] > 0 && G.arcs[v][i] < INT\_MAX) //无穷大宏**

**return i;**

**}**

**return -1;**

**}**

**//创建无向图**

**void CreatUDG(MGraph &G)**

**{**

**int i,j,k,w;**

**char v1,v2;**

**printf("输入顶点数和边数：");**

**scanf("%d%d",&G.vexnum,&G.arcnum);**

**visited = (int \*)malloc(G.vexnum\*sizeof(int));**

**for ( i = 0; i < G.vexnum; i++ )**

**visited[i] = 0;**

**printf("请按次序输入%d个顶点字母标号(如ABC等)：",G.vexnum);**

**getchar();//弹出缓冲区中上次最后出入的换行符，即最后按下的回车键**

**for ( i = 0; i < G.vexnum; i++ )**

**scanf("%c",&G.vexs[i]);**

**getchar();//弹出缓冲区中上次最后出入的换行符，即最后按下的回车键**

**for ( i = 0; i < G.vexnum; ++i )**

**for ( j = 0; j < G.vexnum; ++j )**

**{//初始化邻接矩阵**

**if ( i == j )**

**G.arcs[i][j] = 0;**

**else**

**G.arcs[i][j] = INT\_MAX;**

**}**

**printf("输入边的顶点和权值(如A B 1):\n");**

**for ( k = 0; k < G.arcnum; k++ )**

**{**

**scanf("%c %c %d",&v1,&v2,&w);**

**i = LocateVex(G,v1);**

**j = LocateVex(G,v2);**

**G.arcs[i][j] = w;**

**G.arcs[j][i] = w;**

**getchar();//弹出缓冲区中上次最后出入的换行符，即最后按下的回车键**

**}**

**}**

**//选择创建图的类型**

**void CreatGraph(MGraph &G)**

**{**

**int k;**

**printf("创建有向图:\n");**

**G.kind = UDG;**

**CreatUDG(G);**

**}**

**void DFS(const MGraph &G,int v)**

**{**

**int w;**

**visited[v] = 1;**

**printf("%-4c",G.vexs[v]);**

**for ( w = firstAjdVex(G,v); w >= 0; w = nextAjdVex(G,v,w) )**

**if ( !visited[w] )**

**DFS(G,w);**

**}**

**//深度优先遍历函数**

**void DFSTraverse(const MGraph &G)**

**{**

**int i;**

**for ( i = 0; i < G.vexnum; i++ )**

**visited[i] = 0; //初始化访问标记数组**

**for ( i = 0; i < G.vexnum; i++ )**

**if ( !visited[i] )**

**{//对尚未访问的顶点调用DFS**

**DFS(G,i);**

**}**

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

**}**

**//主函数**

**int main()**

**{**

**MGraph G;**

**CreatGraph(G);**

**//深度优先遍历结果**

**printf("\n深度优先遍历:");**

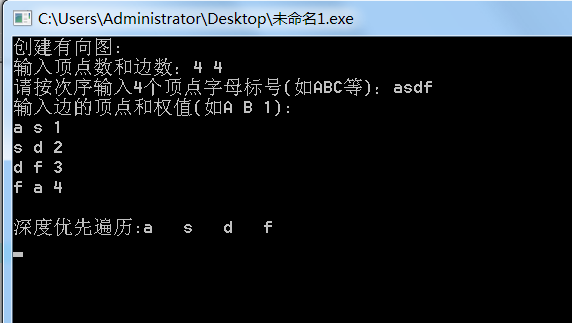
**DFSTraverse(G);**

**getchar();getchar();getchar();getchar();**

**return 0;**

**}**

**六、测试结果：**

****

**七、实验总结：**

**对于深度优先遍历，主要是采用递归的方式。试验本身问题不是太大，但要注意输入的问题，因为一旦数据的输入有问题，结果当然也就不可能正确了。只有正确的输入数据，建立图，才能得出正确的遍历结果。**