

天津商业大学学生实验报告

开课实验室：信息实验室

开课时间 13 年 9 月 1 日

实验报告 13 年 11 月 12 日

学院名称	信息工程	年级、专业、班	软件工程 12-01 班	学号	20125041	姓名	王靖伟	同组姓名	
课程名称	数据结构	实验项目名称	实验四 图的遍历			指导教师	黄橡丽		
实验类型	验证 <input checked="" type="checkbox"/> 综合 <input type="checkbox"/> 设计 <input type="checkbox"/> 创新 <input type="checkbox"/>						成绩		
教师评语	教师签名：_____ 年 月 日								
实验报告内容一般包括以下几个内容：1、目的要求 2、仪器用具及材料（仪器名称及主要规格、用具名称） 3、实验内容及原理（简单但要抓住要点，写出依据原理） 4、操作方法与实验步骤 5、数据图表格（照片） 6、实验过程原始记录 7 数据处理及结果（按实验要求处理数据、结论） 8、作业题 9、讨论（对实验中存在的问题、进一步的想法等进行讨论）									
实验报告内容： 1. 实验目的、要求： （1）掌握图的二种存储结构和二种遍历方法。 （2）学生用 C++/C 完成算法设计和程序设计并上机调试通过。 （3）撰写实验报告，提供实验测试数据和实验结果。 （4）分析算法，要求给出具体的算法分析结果，包括时间复杂度和空间复杂度，并简要给出算法设计小结和心得。 2. 实验内容（一）： （1）实验题目 编程建立图的邻接矩阵存储结构，实现图的深度优先搜索算法。 ① 采用邻接矩阵存储结构创建一个图； ② 图的深度优先搜索算法的实现。 （2）算法设计思想（说明整个程序由一个主函数和哪几个函数组成，并给出主要函数的算法设计思想） 1. 图 G 中查找元素 c 的位置 int Locate(MGraph G,char c) 2. 创建无向图的邻接矩阵存储结构 void CreateUDN(MGraph &G) //接收回车，初始化顶点，接收回车，初始化邻接矩阵，初始化弧，输入一条边依附的顶点和权值 3. 对连通图 G 深度优先遍历 void DFS(MGraph G, int v) //图 G 为邻接矩阵类型，访问第 v 个顶点，依次检查邻接矩阵 v 所在的行，w 是 v 的邻接点，如果 w 未访问，则递归调用 DFS 4. 对非连通图 G 作深度优先遍历 void DFSTraverse(MGraph G, int v) //访问标志数组初始化，对尚未访问的顶点调用 DFS 5. 主函数 void main()									

(3) 程序清单

行号	代码
1	#include "stdio.h"
2	#include "stdlib.h"
3	#define MaxInt 32767
4	#define MAX_VEX 20 //最大顶点个数
5	typedef enum{FALSE,TRUE} boolean;
6	typedef char VertexType;
7	typedef struct{ //图的邻接矩阵存储结构
8	VertexType vexs[MAX_VEX]; //顶点向量
9	int arcs[MAX_VEX][MAX_VEX]; //邻接矩阵
10	int vexnum,arcnum; //图的当前顶点数和弧数
11	}MGraph;
12	boolean visited[MAX_VEX]; //访问标志数组
13	
14	//图 G 中查找元素 c 的位置
15	int Locate(MGraph G,char c){
16	for(int i=0;i<G.vexnum;i++)
17	if(G.vexs[i]==c) return i;
18	return -1;
19	}
20	
21	//创建无向网的邻接矩阵存储结构
22	void CreateUDN(MGraph &G){
23	int i,j,w,s1,s2;
24	char a,b,c,temp;
25	printf("输入顶点数和弧数:");
26	scanf("%d%d",&G.vexnum,&G.arcnum);
27	temp=getchar(); //接收回车
28	printf("输入%d 个顶点 (以空格做间隔): \n",G.vexnum);
29	for(i=0;i<G.vexnum;i++){ //初始化顶点
30	scanf("%c",&G.vexs[i]);
31	temp=getchar(); //接收回车
32	}
33	for(i=0;i<G.vexnum;i++) //初始化邻接矩阵
34	for(j=0;j<G.vexnum;j++)
35	G.arcs[i][j]=MaxInt; //初始化弧
36	printf("输入%d 条弧及其权值 (以空格做间隔): \n",G.arcnum);
37	for(i=0;i<G.arcnum;i++){
38	printf("输入弧%d: ",i+1);
39	scanf("%c %c %d%c",&a,&b,&w,&c); //输入一条边依附的顶点和权值
40	s1=Locate(G,a);
41	s2=Locate(G,b);
42	G.arcs[s1][s2]=G.arcs[s2][s1]=w;
43	}
44	}
45	
46	//对连通图 G 深度优先遍历

```

47 void DFS(MGraph G, int v){                                //图 G 为邻接矩阵类型
48     int w;
49     printf(" -> %c",G.vexs[v]);  visited[v] = TRUE;//访问第 v 个顶点
50     for(w = 0; w< G.vexnum; w++)                          //依次检查邻接矩阵 v 所在的行
51         if((G.arcs[v][w]!=MaxInt)&& (!visited[w]))  DFS(G, w);
52                                     //w 是 v 的邻接点，如果 w 未访问，则递归调用 DFS
53 }
54
55 // 对非连通图 G 作深度优先遍历。
56 void DFSTraverse(MGraph G, int v) {
57     for (v=0; v<G.vexnum; ++v)
58         visited[v] = FALSE;                                // 访问标志数组初始化
59     for (v=0; v<G.vexnum; ++v)
60         if (!visited[v])  DFS(G, v);                       // 对尚未访问的顶点调用 DFS
61 }
62
63 //主函数
64 void main(){
65     MGraph G;
66     CreateUDN (G);
67     printf("深度优先遍历: ");
68     DFSTraverse (G,0);
69     printf("\n");
70 }

```

(3) 测试数据

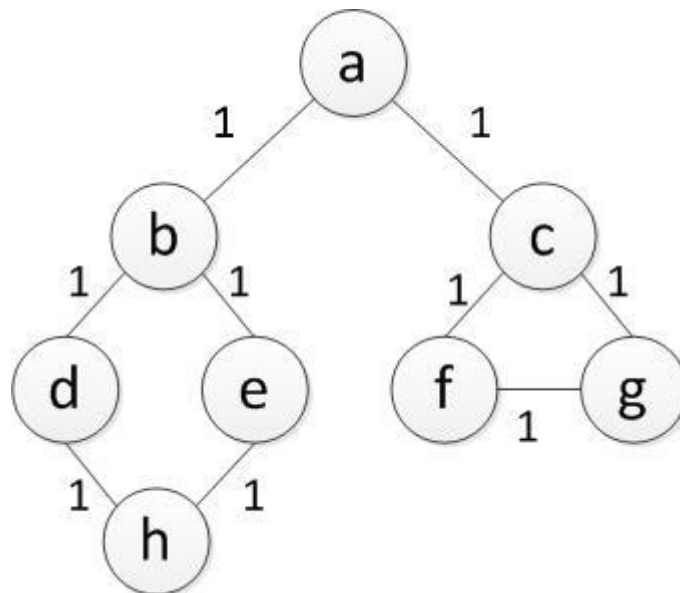


图 1. 深度优先搜索算法的测试用例

测试结果应为：a->b->d->h->e->c->f->g

(4) 实验结果

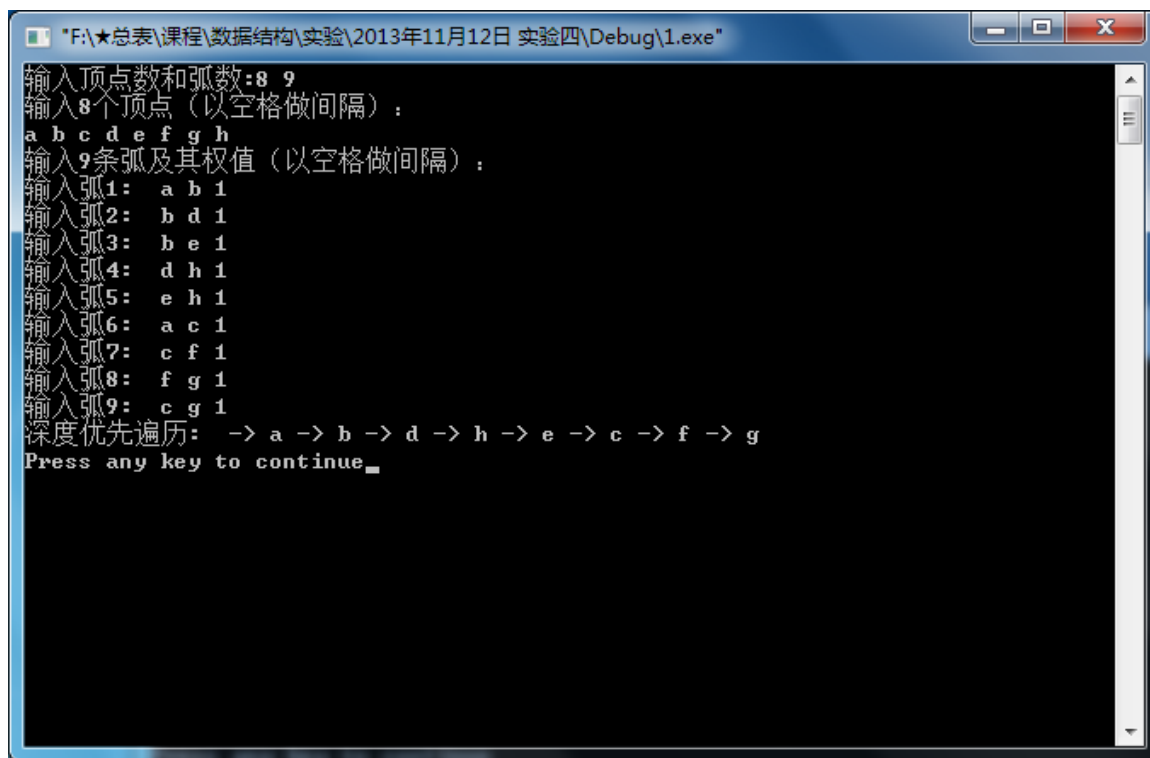


图 2. 深度优先搜索算法的测试结果

(5) 实验收获

分析上述算法，在图遍历时，对图中的，每个顶点至多调用一次 DFS 函数，因为一旦某个顶点被标志成已被访问，就不再从它的出发进行搜索。因此，遍历图的过程实质上是对每个顶点查找其邻接点的过程。其耗费的时间则取决于所采用的存储结构。当二维数组表示邻接矩阵作图的存储结构时，查找每个点的邻接点所需时间为 $O(n^2)$ ，其中 n 为图中顶点数。而当以邻接表作图的存储结构时，找邻接点所需的时间为 $O(e)$ ，其中 e 为无向图中边的数或有向图中弧的数。由此，当以邻接表作存储结构时，深度优先搜索遍历图的时间复杂度为 $O(n+e)$ 。

3. 实验内容 (二):

(1) 实验题目

编程建立图的邻接矩阵存储结构，实现图的广度优先搜索算法。

① 采用邻接矩阵存储结构创建一个图；

② 图的广度优先搜索算法的实现。

(2) 算法设计思想 (说明整个程序由一个主函数和哪几个函数组成，并给出主要函数的算法设计思想)

1. 图 G 中查找元素 c 的位置 `int Locate(MGraph G,char c)`
2. 创建无向网的邻接矩阵存储结构 `void CreateUDN(MGraph &G)`
//接收回车，初始化顶点，接收回车，初始化邻接矩阵，初始化弧，输入一条边依附的顶点和权值
3. 图 G 中顶点 k 的第一个邻接顶点 `int FirstVex(MGraph G,int k)`
4. 图 G 中顶点 i 的第 j 个邻接顶点的下一个邻接顶点 `int NextVex(MGraph G,int i,int j)`
5. 广度优先遍历 `void BFS(MGraph G)`
//辅助队列 Q ， i 尚未访问， i 入列，队头元素出列并置为 k ， w 为 k 的尚未访问的邻接顶点
6. 主函数 `void main()`

(3) 程序清单

行号	代码
1	#include "stdio.h"
2	#include "stdlib.h"
3	#define MaxInt 32767
4	#define MAX_VEX 20 //最大顶点个数
5	#define QUEUE_SIZE (MAX_VEX+1) //队列长度
6	typedef enum {FALSE,TRUE} boolean;
7	typedef char VertexType;
8	typedef struct{ //图的邻接矩阵存储结构
9	VertexType vexs[MAX_VEX]; //顶点向量
10	int arcs[MAX_VEX][MAX_VEX]; //邻接矩阵
11	int vexnum,arcnum; //图的当前顶点数和弧数
12	}MGraph;
13	boolean visited[MAX_VEX]; //访问标志数组
14	class Queue{ //队列类型
15	public:
16	void InitQueue(){
17	base=(int *)malloc(QUEUE_SIZE*sizeof(int));
18	front=rear=0;
19	}
20	void EnQueue(int e){
21	base[rear]=e;
22	rear=(rear+1)%QUEUE_SIZE;
23	}
24	void DeQueue(int &e){
25	e=base[front];
26	front=(front+1)%QUEUE_SIZE;
27	}
28	int *base;
29	int front;
30	int rear;
31	};
32	//图 G 中查找元素 c 的位置
33	int Locate(MGraph G,char c){
34	for(int i=0;i<G.vexnum;i++)
35	if(G.vexs[i]==c) return i;
36	return -1;
37	}
38	//创建无向网的邻接矩阵存储结构
39	void CreateUDN(MGraph &G){
40	int i,j,w,s1,s2;
41	char a,b,c,temp;
42	printf("输入顶点数和弧数:");
43	scanf("%d%d",&G.vexnum,&G.arcnum);
44	temp=getchar(); //接收回车
45	printf("输入%d个顶点 (以空格做间隔): \n",G.vexnum);
46	for(i=0;i<G.vexnum;i++){ //初始化顶点

```

47     scanf("%c",&G.vexs[i]);
48     temp=getchar();                //接收回车
49 }
50 for(i=0;i<G.vexnum;i++)            //初始化邻接矩阵
51     for(j=0;j<G.vexnum;j++)
52         G.arcs[i][j]=MaxInt;
53 printf("输入%d 条弧及其权值（以空格做间隔）：\n",G.arcnum);
54 for(i=0;i<G.arcnum;i++){          //初始化弧
55     printf("输入弧%d:  ",i+1);
56     scanf("%c %c %d%c",&a,&b,&w,&c);    //输入一条边依附的顶点和权值
57     s1=Locate(G,a);
58     s2=Locate(G,b);
59     G.arcs[s1][s2]=G.arcs[s2][s1]=w;
60 }
61 }
62 //图 G 中顶点 k 的第一个邻接顶点
63 int FirstVex(MGraph G,int k){
64     if(k>=0 && k<G.vexnum){ //k 合理
65         for(int i=0;i<G.vexnum;i++)
66             if(G.arcs[k][i]!=MaxInt) return i;
67     }
68     return -1;
69 }
70 //图 G 中顶点 i 的第 j 个邻接顶点的下一个邻接顶点
71 int NextVex(MGraph G,int i,int j){
72     if(i>=0 && i<G.vexnum && j>=0 && j<G.vexnum){ //i,j 合理
73         for(int k=j+1;k<G.vexnum;k++)
74             if(G.arcs[i][k]!=MaxInt) return k;
75     }
76     return -1;
77 }
78 //广度优先遍历
79 void BFS(MGraph G){
80     int k;
81     Queue Q;                //辅助队列 Q
82     Q.InitQueue();
83     for(int v=0;v<G.vexnum;v++)
84         visited[v]=FALSE;
85     for(int i=0;i<G.vexnum;i++)
86         if(!visited[i]){    //i 尚未访问
87             visited[i]=TRUE;
88             printf("%c ",G.vexs[i]);
89             Q.Enqueue(i);    //i 入列
90             while(Q.front!=Q.rear){
91                 Q.DeQueue(k);    //队头元素出列并置为 k
92                 for(int w=FirstVex(G,k);w>=0;w=NextVex(G,k,w))
93                     if(!visited[w]){    //w 为 k 的尚未访问的邻接顶点
94                         visited[w]=TRUE;

```

```

95         printf("-> %c ",G.vexs[w]);
96         Q.Enqueue(w);
97     }
98 }
99 }
100 }
101 //主函数
102 void main(){
103     MGraph G;
104     CreateUDN (G);
105     printf("广深度优先遍历: ");
106     BFS (G);
107     printf("\n");
108 }

```

(3) 测试数据

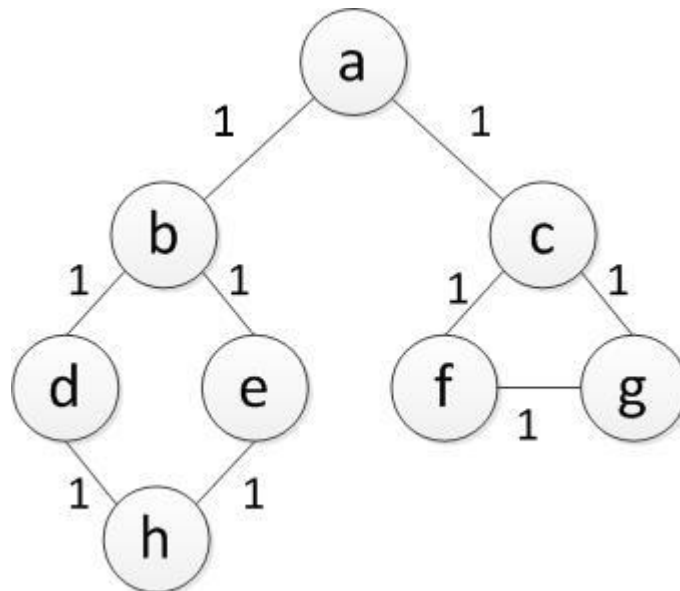
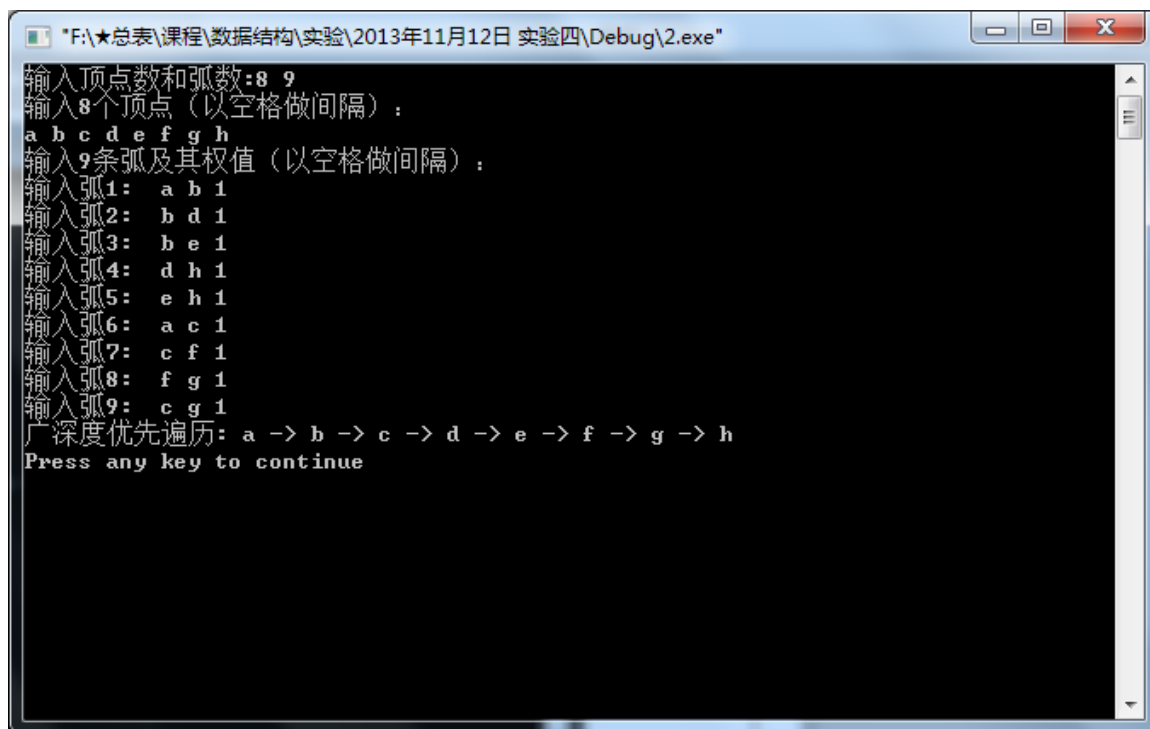


图 3. 广度优先搜索算法的测试用例

测试结果应为：a->b->c->d->e->f->g->h

(4) 实验结果



```
"F:\★总表\课程\数据结构\实验\2013年11月12日 实验四\Debug\2.exe"
输入顶点数和弧数:8 9
输入8个顶点 (以空格做间隔):
a b c d e f g h
输入9条弧及其权值 (以空格做间隔):
输入弧1: a b 1
输入弧2: b d 1
输入弧3: b e 1
输入弧4: d h 1
输入弧5: e h 1
输入弧6: a c 1
输入弧7: c f 1
输入弧8: f g 1
输入弧9: c g 1
广深度优先遍历: a -> b -> c -> d -> e -> f -> g -> h
Press any key to continue
```

图 4. 广度优先搜索算法的测试结果

(5) 实验收获

分析上述算法，每个顶点至多进一次队列。遍历图的过程实质上是通过边或弧找邻接点的过程，因此广度优先搜索遍历图的时间复杂度和深度优先搜索遍历相同，两者不同之处仅仅在于对顶点的访问顺序不同。

注 1. 每个实验项目一份实验报告。2. 实验报告第一页学生必须使用规定的实验报告纸书写，附页用实验报告附页纸或 A4 纸书写，字迹工整，曲线要画在坐标纸上，线路图要整齐、清楚（不得徒手画）。3. 实验教师必须对每份实验报告进行批改，用红笔指出实验报告中的错、漏之处，并给出评语、成绩，签全名、注明日期。4. 待实验课程结束以后，要求学生把实验报告整理好，交给实验指导教师，加上实验课学生考勤及成绩登记表（见附件 2）、目录和学院统一的封面（见附件 3）后，统一装订成册存档。

制表单位：设备处