**Project 9 DFS与BFS**

学号 2014211116 姓名 骆金参 上交时间 2015-12-6

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 题号 | 程序逻辑(40) | 算法新颖性(20) | 代码规范  (20) | 实验报告(20) | 总分 |
| 得分 |  |  |  |  |  |

1. **问题描述**

写一程序实现图的深度优先搜索算法（DFS），并对输入的有向图输出遍历结果。

写一程序实现图的广度优先搜索算法（BFS），并对输入的有向图输出遍历结果。

**2. 算法描述**

**#include <iostream>**

**#include <malloc.h>**

**#define INFINITY 32767**

**#define MAX\_VEX 20 //最大顶点个数**

**#define QUEUE\_SIZE (MAX\_VEX+1) //队列长度**

**using namespace std;**

**bool \*visited; //访问标志数组**

**//图的邻接矩阵存储结构**

**typedef struct{**

**char \*vexs; //顶点向量**

**int arcs[MAX\_VEX][MAX\_VEX]; //邻接矩阵**

**int vexnum,arcnum; //图的当前顶点数和弧数**

**}Graph;**

**//队列类**

**class Queue{**

**public:**

**void InitQueue() {**

**base=(int \*)malloc(QUEUE\_SIZE\*sizeof(int));**

**front=rear=0;**

**}**

**void EnQueue(int e) {**

**base[rear]=e;**

**rear=(rear+1)%QUEUE\_SIZE;**

**}**

**void DeQueue(int &e) {**

**e=base[front];**

**front=(front+1)%QUEUE\_SIZE;**

**}**

**public:**

**int \*base;**

**int front;**

**int rear;**

**};**

**//图G中查找元素c的位置**

**int Locate(Graph G,char c) {**

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

**if(G.vexs[i]==c) return i;**

**return -1;**

**}**

**//创建无向网**

**void CreateUDN(Graph &G) {**

**int i,j,w,s1,s2;**

**char a,b,temp;**

**printf("输入顶点数和弧数:");**

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

**temp=getchar(); //接收回车**

**G.vexs=(char \*)malloc(G.vexnum\*sizeof(char)); //分配顶点数目**

**printf("输入%d个顶点.\n",G.vexnum);**

**for(i=0;i<G.vexnum;i++){ //初始化顶点**

**printf("输入顶点%d:",i);**

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

**temp=getchar(); //接收回车**

**}**

**for(i=0;i<G.vexnum;i++) //初始化邻接矩阵**

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

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

**printf("输入%d条弧.\n",G.arcnum);**

**for(i=0;i<G.arcnum;i++){ //初始化弧**

**printf("输入弧%d:",i);**

**scanf("%c %c %d",&a,&b,&w); //输入一条边依附的顶点和权值**

**temp=getchar(); //接收回车**

**s1=Locate(G,a);**

**s2=Locate(G,b);**

**G.arcs[s1][s2]=G.arcs[s2][s1]=w;**

**}**

**}**

**//图G中顶点k的第一个邻接顶点**

**int FirstVex(Graph G,int k) {**

**if(k>=0 && k<G.vexnum) { //k合理**

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

**if(G.arcs[k][i]!=INFINITY) return i;**

**}**

**return -1;**

**}**

**//图G中顶点i的第j个邻接顶点的下一个邻接顶点**

**int NextVex(Graph G,int i,int j){**

**if(i>=0 && i<G.vexnum && j>=0 && j<G.vexnum) { //i,j合理**

**for(int k=j+1;k<G.vexnum;k++)**

**if(G.arcs[i][k]!=INFINITY) return k;**

**}**

**return -1;**

**}**

**//深度优先遍历**

**void DFS(Graph G,int k) {**

**int i;**

**if(k==-1) { //第一次执行DFS时,k为-1**

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

**if(!visited[i]) DFS(G,i); //对尚未访问的顶点调用DFS**

**}**

**else {**

**visited[k]=true;**

**printf("%c ",G.vexs[k]); //访问第k个顶点**

**for(i=FirstVex(G,k);i>=0;i=NextVex(G,k,i))**

**if(!visited[i]) DFS(G,i); //对k的尚未访问的邻接顶点i递归调用DFS**

**}**

**}**

**//广度优先遍历**

**void BFS(Graph G){**

**int k;**

**Queue Q; //辅助队列Q**

**Q.InitQueue();**

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

**if(!visited[i]) { //i尚未访问**

**visited[i]=true;**

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

**Q.EnQueue(i); //i入列**

**while(Q.front!=Q.rear) {**

**Q.DeQueue(k); //队头元素出列并置为k**

**for(int w=FirstVex(G,k);w>=0;w=NextVex(G,k,w))**

**if(!visited[w]){ //w为k的尚未访问的邻接顶点**

**visited[w]=true;**

**printf("%c ",G.vexs[w]);**

**Q.EnQueue(w);**

**}**

**}**

**}**

**}**

**//主函数**

**int main() {**

**int i;**

**Graph G;**

**CreateUDN(G);**

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

**printf("\n广度优先遍历: ");**

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

**visited[i]=false;**

**DFS(G,-1);**

**printf("\n深度优先遍历: ");**

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

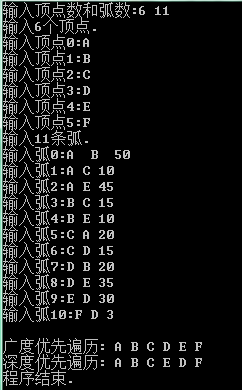
**visited[i]=false;**

**BFS(G);**

**return 0;**

**}**

**3.测试结果**



**4. 分析与评论**

不懂，要求的没做完，没有指定起始的顶点，文件读取与输出也没做。。。