AI实验一，深度优先遍历和广度优先遍历

#include"stdio.h"

#include <queue>

using namespace std;

#define MaxVnum 50

typedef double AdjMatrix[MaxVnum][MaxVnum];

typedef struct {

int vexnum,arcnum;

AdjMatrix arcs;

}Graph;

void CreateGraph(Graph &G)

{

G.vexnum=8;

G.arcnum=9;

G.arcs[0][1]=1;

G.arcs[0][2]=1;

G.arcs[1][3]=1;

G.arcs[1][4]=1;

G.arcs[2][5]=1;

G.arcs[2][6]=1;

G.arcs[3][1]=1;

G.arcs[3][7]=1;

G.arcs[3][6]=1;

G.arcs[4][1]=1;

G.arcs[4][7]=1;

G.arcs[5][2]=1;

G.arcs[5][6]=1;

G.arcs[5][5]=1;

G.arcs[6][2]=1;

G.arcs[6][5]=1;

G.arcs[7][3]=1;

G.arcs[7][4]=1;

}//建表

bool visited[MaxVnum];

void DFS(Graph G,int v)

{

int a;

a=v;

visited[a]= true;

printf("%d",a);

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

if(G.arcs[a][j]==1&&visited[j]== false)

DFS(G,j);

}//深度优先遍历

/\*void DFSTraverse(Graph G) {

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

visited[v] = false;

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

if (visited[v] == false)

DFS(G, v);

}\*/

void BFSTraverse(Graph G)

{

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

visited[v]=false;

queue<int> Q;

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

{

if(visited[v]==false)

{

Q.push(v);

visited[v]=true;

while (!Q.empty())

{

int t =Q.front();

Q.pop();

printf(" %d ",t+1);

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

if(G.arcs[t][j]==1&&visited[j]== false)

{

Q.push(j);

visited[j]=true;

}

}

}

}

}//广度优先遍历

int main()

{

int i,j,a=0;

Graph G;

CreateGraph(G);

for(i=0;i<7;i++)

{

for(j=0;j<7;j++)

{

printf(" %0.0lf ",G.arcs[i][j]);

}

printf("\n");

}

DFS(G,a);

printf("\n");

/\*

DFSTraverse(G);

printf("\n");

\*/

BFSTraverse(G);

printf("\n");

return 0;

}

