【邻接矩阵的深度优先遍历】

/\*\*

\* AdList\_DFS.c

\* 邻接矩阵实现 图的深度遍历

\* @author Darbuly 2018-2019

\* @copyright 2018-2019 Darbuly

\*/

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#define VRType int

#define InfoType int

#define VertexType char

#define MAX\_VERTEX\_NUM 20

#define MAX\_VALUE 0

typedef enum{DG,DN,AG,AN}GraphKind;

typedef struct ArcCell{

VRType adj;

InfoType \*info;

}ArcCell,AdjMatrix[MAX\_VERTEX\_NUM][MAX\_VERTEX\_NUM];

typedef struct{

VertexType vexs[MAX\_VERTEX\_NUM];

AdjMatrix arc;

int vernum,arcnum;

GraphKind kind;

}MGraph;

void g\_create(MGraph \*graph)

{

int num;

int i,j,k;

char c;

printf("Please enter the num of vertex:\n");

scanf("%d",&graph->vernum);

getchar();

printf("Please enter vertex infomations:\n");

for(i=0;i<graph->vernum;i++)

{

scanf("%c",&graph->vexs[i]);

getchar();

}

for(i=0;i<graph->vernum;i++)

for(j=0;j<graph->vernum;j++)

graph->arc[i][j].adj = MAX\_VALUE;

graph->arcnum = 0;

for(i=0;i<graph->vernum;i++)

{

printf("Please enter vertex nextto the %c ,and end by #\n",graph->vexs[i]);

for(j=0;j<graph->vernum;j++)

{

scanf("%c",&c);

if(c=='#')

{

getchar();

break;

}

//scanf("%d",&num);

for(k=0;k<graph->vernum;k++)

{

if(graph->vexs[k]!=c)

continue;

//graph->arc[i][k].adj=num;

graph->arc[i][k].adj=1;

graph->arcnum++;

}

getchar();

}

}

graph->arcnum /=2;

printf("\n");

for(i=0;i<graph->vernum;i++)

{

for(j=0;j<graph->vernum;j++)

{

printf("%d ",graph->arc[i][j].adj);

}

printf("\n");

}

}

static void dfs\_graph(MGraph \*graph,bool visited[],const int i);

void g\_depth\_first\_search(MGraph \*graph)

{

bool visited[graph->vernum];

int i;

for(i=0;i<graph->vernum;i++)

visited[i]=false;

visited[0]=true;

dfs\_graph(graph,visited,0);

printf("\n");

}

static void dfs\_graph(MGraph \*graph,bool visited[],const int i)

{

int j;

printf("%c\t",graph->vexs[i]);

for(j=0;j<graph->vernum;j++)

{

if(graph->arc[i][j].adj != MAX\_VALUE && !visited[j])

{

visited[j]=true;

dfs\_graph(graph,visited,j);

}

}

}

int main()

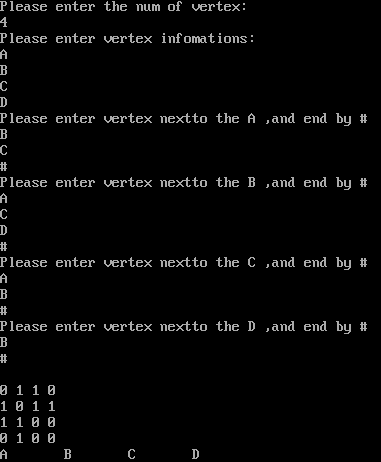
{ MGraph graph;

g\_create(&graph);

g\_depth\_first\_search(&graph);

return 0;

}



代码分析：

时间复杂度：O(n^2)

static void dfs\_graph(MGraph \*graph,bool visited[],const int i)

{

int j;

printf("%c\t",graph->vexs[i]);

for(j=0;j<graph->vernum;j++)

{

if(graph->arc[i][j].adj != MAX\_VALUE && !visited[j])

{

visited[j]=true;

dfs\_graph(graph,visited,j);

}

}

}

矩阵包含n^2个元素，在算法中，共n个顶点，对每个顶点都要遍历n次，所以时间复杂度为O(n^2)

代码思维：

1. 体现的就是深度优先遍历的思维
2. 通过递归的方法来实现，递归的跳出条件就是遍历的次数限制，加上一个if的判断
3. 通过visited探访记录的解决方法，控制递归

改进：

1.