/\*\*

\* AdMatrix.c 邻接矩阵的广度优先遍历

\*

\* @author Darbuly 2018-2019

\* @copylight 2019-2019 Darbuly

\*

\*

\*/

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#define OK 1

#define ERROR 0

#define VRType int

#define InfoType int

#define VertexType char

#define MAX\_VERTEX\_NUM 20

#define MAX\_VALUE 0

typedef int ElemType;

typedef int Status;

int visited[MAX\_VERTEX\_NUM];

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

typedef struct QNode

{

ElemType data;

struct QNode \*next;

}QNode,\*QueuePtr;

typedef struct

{

QueuePtr front,rear;

}LinkQueue;

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;

Status InitQueue(LinkQueue \*q)

{

q->front=q->rear=(QueuePtr)malloc(sizeof(QNode));

if(!q->front)return ERROR;

q->front->next=NULL;

return OK;

}

Status InsertQueue(LinkQueue \*q,ElemType e)

{

QueuePtr p;

p = (QueuePtr)malloc(sizeof(QNode));

if(p==NULL) return ERROR;

p->data=e;

p->next=NULL;

q->rear->next=p;

q->rear = p;

return OK;

}

Status EnQueue(LinkQueue \*q, ElemType \*e)

{

QueuePtr p;

if(q->front == q->rear)return ;

p=q->front->next;

\*e = p->data;

q->front->next = p->next;

if(q->rear=p)q->rear=q->front;

free(p);

return OK;

}

bool QueueEmpty(LinkQueue q)

{

if(q.front==q.rear)return true;

return false;

}

Status DestroyQueue(LinkQueue \*q)

{

while(q->front)

{

q->rear = q->front->next;

free(q->front);

q->front = q->rear;

}

return OK;

}

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;

}

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

{

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

continue;

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

graph->arcnum++;

}

getchar();

}

}

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");

}

}

void BFSTraverse(MGraph graph)

{

int i,j;

LinkQueue Q;

for(i=0;i<graph.vernum;i++)

{

visited[i]=false;

}

InitQueue(&Q);

for(i=0;i<graph.vernum;i++)

{

if(!visited[i])

{

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

visited[i]=true;

InsertQueue(&Q,i);

while(!QueueEmpty(Q))

{

EnQueue(&Q,&i);

for(j=0;j<graph.vernum;j++)

{

if(graph.arc[i][j].adj==1 && !visited[j])

{

printf("%c\t",graph.vexs[j]);

visited[j]=true;

InsertQueue(&Q,j);

}

}

}

}

}

}

int main()

{

LinkQueue q;

MGraph graph;

InitQueue(&q);

DestroyQueue(&q);

g\_create(&graph);

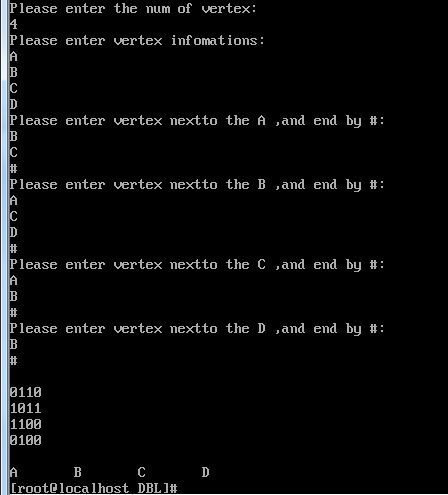
printf("\n");

BFSTraverse(graph);

printf("\n");

return 0;

}



代码分析：

1. 时间复杂度：

假设有n个元素，那么， 每个元素要遍历n次，所以需要遍历n\*n次，所以时间复杂度是

O(n^2)