/\*\*

\* AdList\_BFS.c 邻接表的广度优先遍历

\*

\* @author Darbuly 2018-2019

\* @copylight 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

int visited[MAX\_VERTEX\_NUM];

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

typedef struct ArcNode

{

int adjvex;

struct ArcNode \*nextarc;

}ArcNode;

typedef struct VNode

{

VertexType data;

ArcNode \*firstarc;

}VNode,AdjList[MAX\_VERTEX\_NUM];

typedef struct

{

AdjList vertices;

int vexnum,arcnum;

}ALGraph;

void InsertNode(ALGraph \*G,int i,int j)

{

printf("\ni=%d,j=%d\n",i,j);

ArcNode \*ptrArcNode = (ArcNode\*)malloc(sizeof(ArcNode));

ptrArcNode->adjvex = j;

ptrArcNode->nextarc = G->vertices[i].firstarc;

G->vertices[i].firstarc = ptrArcNode;

printf("%c-%c\n",G->vertices[i].data,G->vertices[j].data);

ptrArcNode = (ArcNode\*)malloc(sizeof(ArcNode));

ptrArcNode->adjvex = i;

ptrArcNode->nextarc = G->vertices[j].firstarc;

G->vertices[j].firstarc = ptrArcNode;

}

void CreateALGraph(ALGraph \*G)

{

int i,j,k;

if(G==NULL)G = (ALGraph\*)malloc(sizeof(ALGraph));

printf("Please input the vexnum and arcnum:\n");

scanf("%d%d",&G->vexnum,&G->arcnum);

getchar();

printf("Please input the Vertices:\n");

for(i=0;i<G->vexnum;i++)

{

printf("Vertex %d:",i+1);

scanf("%c",&G->vertices[i].data);

G->vertices[i].firstarc=NULL;

getchar();

}

for(k=0;k<G->vexnum;k++)

{

printf("Please input (vi,vj) vertex:\n");

scanf("%d%d",&i,&j);

InsertNode(G,i,j);

}

}

void ShowALGraph(ALGraph \*G)

{

ArcNode \*ptrArcNode;

int i;

for(i=0;i<G->vexnum;i++)

{

printf("The Vertex %d is %c :",i+1, G->vertices[i].data);

ptrArcNode = G->vertices[i].firstarc;

while(1)

{

if(ptrArcNode==NULL)break;

printf("->%c",G->vertices[ptrArcNode->adjvex]);

if(ptrArcNode->nextarc==NULL)break;

ptrArcNode = ptrArcNode->nextarc;

}

printf("\n");

}

}

void BFSTraverse(ALGraph G)

{

int i;

ArcNode \*ptrArcNode;

printf("\n");

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

{

visited[i]=false;

}

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

{

if(!visited[i])

{

printf("%c\t",G.vertices[i]);

visited[i]=true;

ptrArcNode = G.vertices[i].firstarc;

while(ptrArcNode->nextarc!=NULL)

{

if(!visited[ptrArcNode->adjvex])

{

printf("%c\t",G.vertices[ptrArcNode->adjvex].data);

visited[ptrArcNode->adjvex]=true;

}

ptrArcNode = ptrArcNode->nextarc;

}

}

}

printf("\n");

}

int main()

{

ALGraph G;

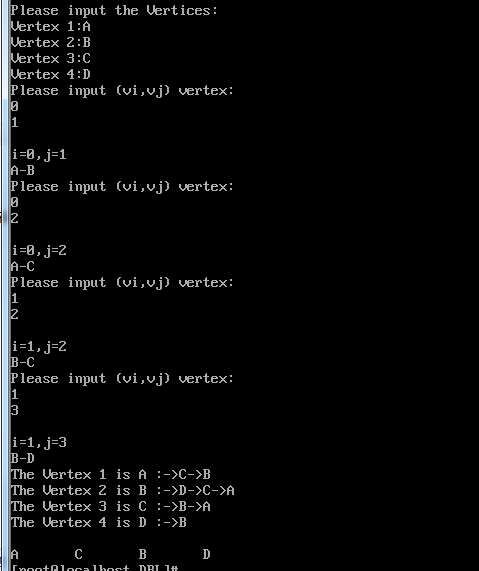
CreateALGraph(&G);

ShowALGraph(&G);

BFSTraverse(G);

return 0;

}



代码分析:

1. 时间复杂度：