#include <stdio.h>

#include <stdlib.h>

#define MAX\_VERTEX\_NUM 50

typedef char VertexData;//顶点数据类型

typedef int WeightType;//权值类型

typedef struct

{

VertexData vers[MAX\_VERTEX\_NUM];

WeightType arcs[MAX\_VERTEX\_NUM][MAX\_VERTEX\_NUM];//权值类型

int vernum;//顶点个数

int arcnum;//边的个数

int kind; //0 direct 1 undirect

}AdjMatrix;

typedef struct

{

int data[MAX\_VERTEX\_NUM];

int front;

int rear;

}Queue;

void InitQueue(Queue\* q)

{

q->front = 0;

q->rear = 0;

}

void enQueue(Queue \*q,int v)

{

q->rear = (q->rear + 1) % MAX\_VERTEX\_NUM;

q->data[q->rear] = v;

}

void deQueue(Queue \*q, int\* v)

{

q->front = (q->front + 1) % MAX\_VERTEX\_NUM;

\*v = q->data[q->rear];

}

bool empty(Queue \*q)

{

return q->front == q->rear;

}

int locateVertex(AdjMatrix \*G,VertexData v)

{

//v 存在返回下标 不存在返回1

int i;

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

{

if(G->vers[i] == v)

{

break;

}

}

if(i < G->vernum)

{

return i;

}

else{

return -1;

}

}

void createAdjGraph(AdjMatrix \*G)

{

int i,j;

char v1,v2;

int w,k;

//v 存在返回下标 不存在返回1

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

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

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

//(2)初始化邻接矩阵

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

{

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

{

if(i==j)

{

G->arcs[i][j] = 0;

}

else{

G->arcs[i][j] = 88888; //最大值

}

}

}

//(3)读入顶点个数

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

{

scanf("%c",&(G->vers[i]));

}

//(4)读入边的个数

//AB56 ('A','B','56')

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

{

scanf("%c%c%d",&v1,&v2,&w);

i = locateVertex(G,v1);

j = locateVertex(G,v2);

if(i != -1 && j != -1)

{

G->arcs[i][j] = w;

if(G->kind == 1)

{

G->arcs[j][i] = w;

}

}

}

}

//深度优先遍历的递归程序

void rdfs(AdjMatrix \*G,int v,bool \*visited)

{

int k;

printf("%c ",G->vers[v]);

visited[v] = true;

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

{

if(G->arcs[v][k] != 0 && G->arcs[v][k] != 88888)

{

if(!visited[k])

{

rdfs(G,k,visited);

}

}

}

}

//深度优先遍历的主程序

void dfs(AdjMatrix \*G)

{

int k;

bool visited[MAX\_VERTEX\_NUM];

//获取状态数组并初始化

for (int i = 0; i < MAX\_VERTEX\_NUM; i++)

visited[i] = false;

//检查未被访问过的顶点

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

{

if(!visited[k])

{

rdfs(G, k, visited);

}

}

}

//广度优先遍历的递归程序

void bfs(AdjMatrix \*G,int v0,bool \*visited)

{

int k,v;

Queue qu;

InitQueue(&qu);

printf("%c ",G->vers[v0]);

visited[v0] = true;

enQueue(&qu,v0);

while(!empty(&qu))

{

deQueue(&qu,&v);

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

{

if(G->arcs[v][k] !=88888 && G->arcs[v][k] !=0)

{

if(!visited[k])

{

printf("%c ",G->vers[k]);

visited[k] = true;

enQueue(&qu,k);

}

}

}

}

}

//广度优先遍历的主程序

void bfs(AdjMatrix \*G)

{

int k;

bool visited[MAX\_VERTEX\_NUM];

//获取状态数组并初始化

visited[k] = false;

//检查未被访问过的顶点

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

{

if(!visited[k])

{

bfs(G, k, visited);

}

}

}

//1 5 5ABCDEAB1AC2AD3CD5DE4

//1 5 5ACBDEAB1AC2AD3CD5DE4

int main(){

AdjMatrix graph;

createAdjGraph(&graph);

printf("深度优先遍历的顺序");

dfs(&graph);

printf("\n");

printf("广度优先遍历的顺序");

bfs(&graph);

printf("\n");

return 1;

}