设计算法，打印连通图G中每个顶点一次且仅一次，并要求打印次序满足以下条件：距离顶点v0近的顶点先于距离远的顶点（以边数为单位）。

思路：假设图G是无向图，建立好无向图后bfs（相当于二叉树的层序遍历，就是一一个点为中心，遍历他周围的节点）就能按距离顶点近的顶点先于距离远的顶点的次序打印连通图G中每个顶点一次且仅一次。

输入：

第一行输入两个数n，m，n表示无向图的点的数量，m表示输入m条边的信息。

接下来的m行，每行输入i和j，表示点i和点j之间有一条无向边。

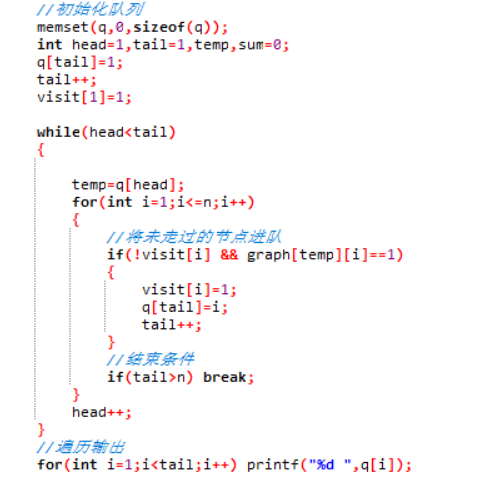
输出：

输出为一行，按距离顶点近的顶点先于距离远的顶点的次序打印连通图G中每个顶点一次且仅一次，点与点之间用空格分隔。

算法：

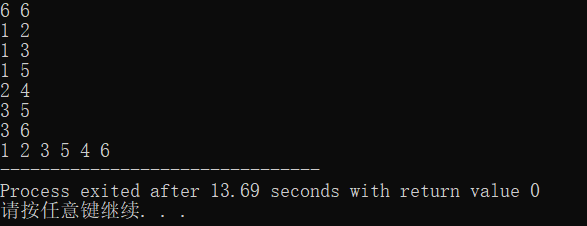
用q[]数组来存储bfs过程中的经过的点的顺序，用visit[]数组来记录是否访问过点，visit[i]=1表示访问过，否则没有。用head，tail配合q[]数组来模拟一个队列。从head处开始循环，将未经过的点进队。最后遍历输出q[]数组。

代码如下：



附件：

1. 示例无向图
2. 程序截图



1. 源代码

#include<bits/stdc++.h>

using namespace std;

const int INF=0x3f;

#define max 101

int graph[max][max],visit[max],q[max];

int n,m;

int main()

{

memset(visit,0,sizeof(visit)); //初始化标记数组

cin>>n>>m;

//初始化无向图

for(int i=1;i<=n;i++)

{

for(int j=1;j<=n;j++)

if(i==j) graph[i][j]=0;

else graph[i][j]=INF;

}

//建立无向图

while(m--)

{

int u,v;

cin>>u>>v;

graph[v][u]=1;

graph[u][v]=1;

}

//初始化队列

memset(q,0,sizeof(q));

int head=1,tail=1,temp;

q[tail]=1;

tail++;

visit[1]=1;

while(head<tail)

{

temp=q[head];

for(int i=1;i<=n;i++)

{

//将未走过的节点进队

if(!visit[i] && graph[temp][i]==1)

{

visit[i]=1;

q[tail]=i;

tail++;

}

//结束条件

if(tail>n) break;

}

head++;

}

//遍历输出

for(int i=1;i<tail;i++) printf("%d ",q[i]);

return 0;

}