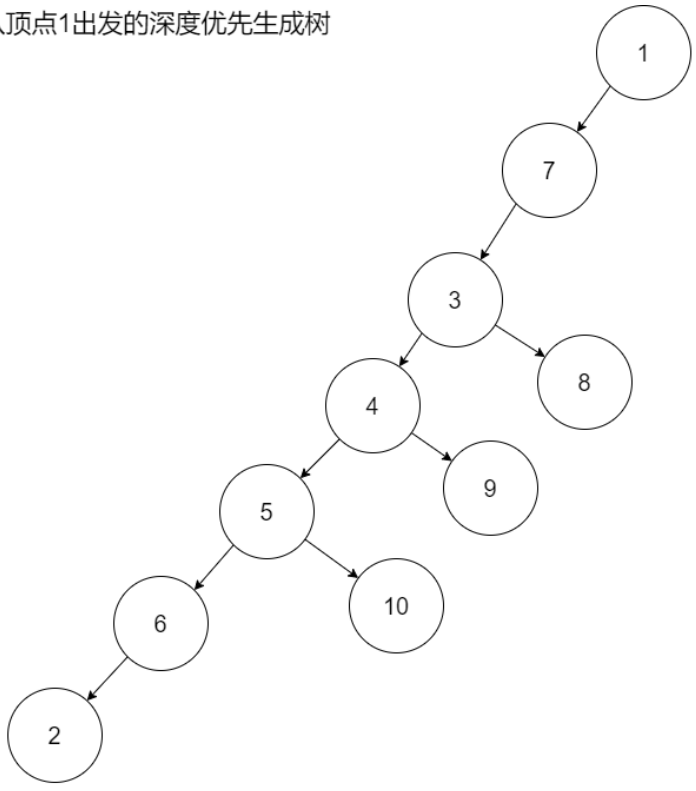
1、 已知二维数组表示的图的邻接矩阵如下图所示，分别画出自顶点 1 出发进行遍历所得的深度优先生成树和广度优先生成树。

	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	1	0	1	0
2	0	0	1	0	0	0	1	0	0	0
3	0	0	0	1	0	0	0	1	0	0
4	0	0	0	0	1	0	0	0	1	0
5	0	0	0	0	0	1	0	0	0	1
6	1	1	0	0	0	0	0	0	0	0
7	0	0	1	0	0	0	0	0	0	1
8	1	0	0	1	0	0	0	0	1	0
9	0	0	0	0	1	0	1	0	0	1
10	1	0	0	0	0	1	0	0	0	0

解：

从顶点1出发的深度优先生成树



1 → 7 → 3 → 4 → 5 → 6 → 2 → 10 → 9 → 8

Figure1.1 从顶点 1 出发生成的深度优先生成树

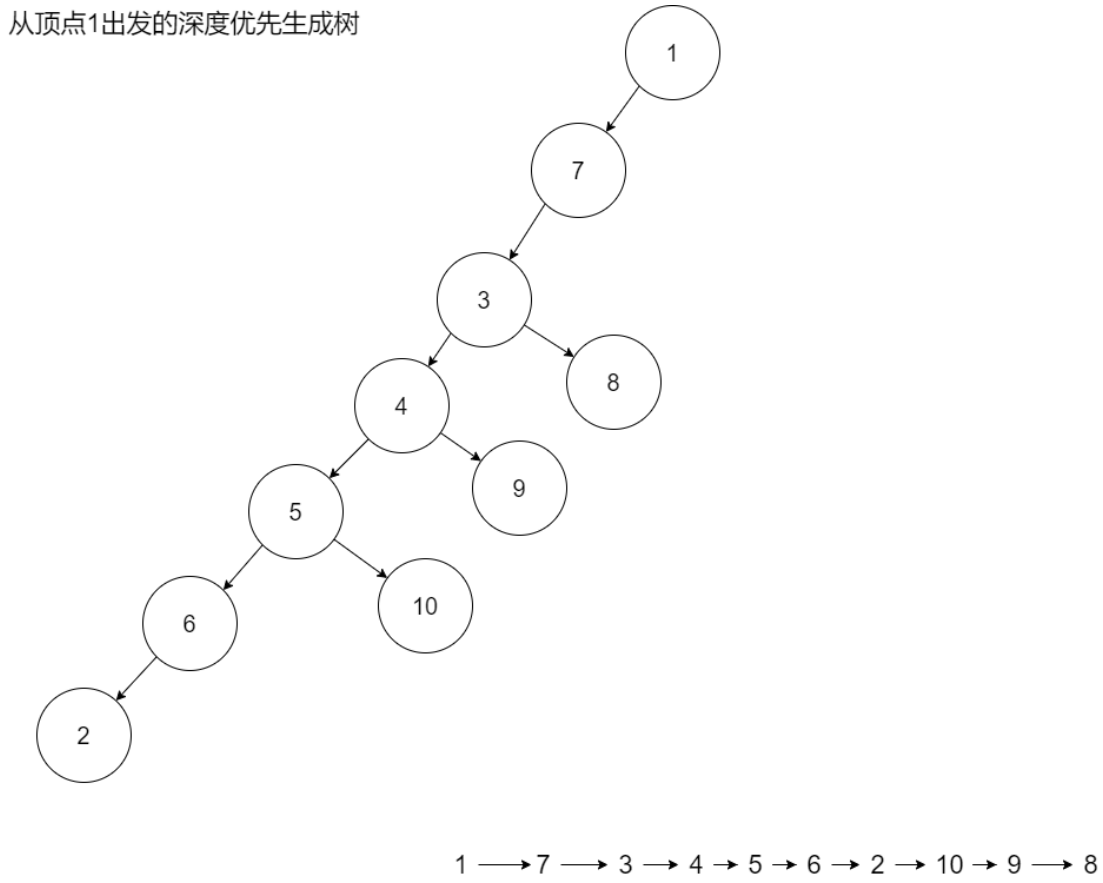


Figure1.2 从顶点 1 出发的广度优先生成树

2、 设计一个算法,判断一个未知顶点个数和边数的无向连通图 G 是否是棵树,假设图采用邻接表存储。若是树,返回 true;否则返回 false。
(用图 1 和图 2 验证作业题 2 算法的正确性)

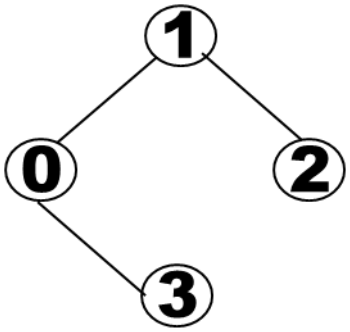


图1

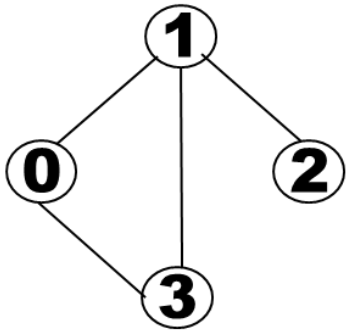


图2

解：该题的完整代码在 `Code` 文件夹下（HW11.2.cpp\HW11.1.exe）
运行结果：

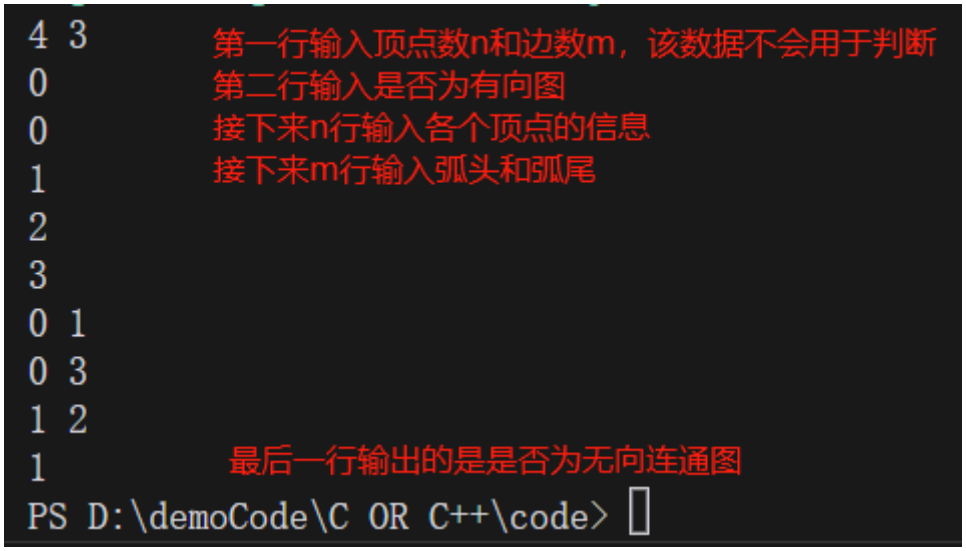


Figure2.1 图一的运行结果（为无向连通图）

Figure2.2 图二的运行结果（不为无向连通图）

主控函数:

bool IsTree(Graph G){ }

```cpp
bool IsTree(Graph G)
{
    bool IsVisited[MaxSize]; // 记录每个顶点是否被访问过
    int NodeCnt = 0;         // 记录访问的顶点个数
    int ArcCnt = 0;          // 记录边的个数
    bool IsTree = false;     // 判断图G是否为树
    DFS(G, G.verList[0], 0, IsVisited, NodeCnt, ArcCnt);
    if ((!G.isDireted) && (ArcCnt / 2 == NodeCnt - 1)) // 因为是无向图所以边要除以
2   {
        IsTree = true; // 无向连通图G中恰好有n-1条边
    }
    return IsTree;
}

```

Figure2.1 IsTree 函数代码快照（原图片在 CodeSnap 文件夹下）

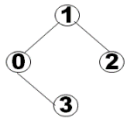
核心函数:

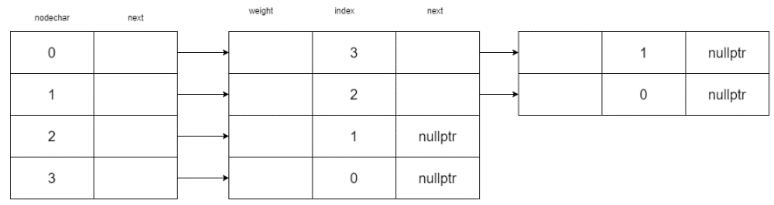
void DFS(Graph G, HNode V, int i, bool IsVisited[], int &NodeCnt, int &ArcCnt){ }

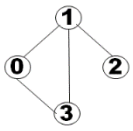
```cpp
void DFS(Graph G, HNode V, int i, bool IsVisited[], int &NodeCnt, int &ArcCnt)
{
    if (!IsVisited[i])
    {
        IsVisited[i] = true;
        NodeCnt++;
        Node *tmp = V.next; // tmp指向HNode的第一个Node结点
        if (tmp == nullptr) // HNode的next为空
        {
            DFS(G, G.verList[i + 1], i + 1, IsVisited, NodeCnt, ArcCnt);
        }
        else // HNode的next不为空
        {
            do
            {
                if (IsArcExist(G.verList[i].nodeData, G.verList[tmp->index].nodeData, G)) // 判断是否有该边长
                {
                    ArcCnt++;
                }
                DFS(G, G.verList[tmp->index], tmp->index, IsVisited, NodeCnt, ArcCnt); // 递归到下一层
                tmp = tmp->next;                                                       // tmp向后移
            } while (tmp != nullptr);                                                  // 当tmp不为nullptr时一直进行DFS并且tmp后
移      }
    }
}
```

Figure2.2 DFS 函数代码快照（原图片在 CodeSnap 文件夹下）

图一图二的邻接表结构：







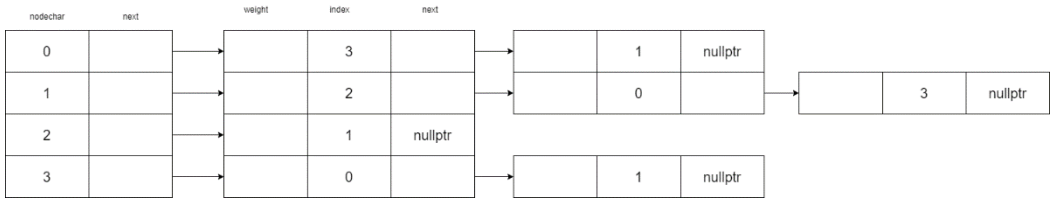


Figure2.3 图一图二邻接表的结构图（原图片在 Photo 文件夹下）

----------END----------