1.证明下列结论：

1）在一个无向图中，如果每个顶点的度大于等于 2，则该图一定含有圈；

2）在一个有向图 D 中，如果每个顶点的出度都大于等于 1，则该图一定含有一个有向圈。

1）证明：设无向图最长的迹P为$\{V_0, V_1, \dots, V_k\}$，每个顶点的度都大于等于 2，故对于顶点V_0来说，存在V'与V_0相邻，若$V' \in P$，则显然存在一个环，若$V' \notin P$，则可以构成一条更长的迹$\{V', V_0, \dots, V_k\}$，这与假设矛盾，故该图一定含有一个环。

2）证明：设有向图最长的有向迹P为$\{V_0, \dots, V_k\}$，又每个顶点的出度都大于等于 1，故存在V'与V_k相邻，若$V' \in P$，则显然存在一个有向环；若$V' \notin P$，则可以构成一条更长的迹$\{V_0, \dots, V_k, V'\}$，这与假设矛盾，故该有向图一定含有一个有向环。

2. 设 D 是至少有三个顶点的连通有向图。如果 D 中包含有向的 Euler 环游（即是通过 D 中每条有向边恰好一次的闭迹），则 D 中每一顶点的出度和入度相等。反之，如果 D 中每一顶点的出度与入度都相等，则 D 一定包含有向的 Euler 环游。这两个结论是正确的吗？请说明理由。如果 G 是至少有三个顶点的无向图，则 G 包含 Euler 环游的条件是什么？

a.设图D中含有 Euler 环游，那么该环游必经过每个顶点至少一次，且每次经过必然伴随着一次进与一次出，故对于每个顶点而言，入度必定等于出度。

b.设D中每个顶点的入度和出度相等，当顶点个数为 2 时，因为每个点的入度和出度相等，故构成 Euler 环游，假设当顶点个数为k时结论成立，则当顶点个数为$k+1$，时，任取 D 中的顶点$v \in v(D)$，设A为以v为终点的边的集合，B是以v为起点的边的集合，由于v的出度等于入度，因此A和B中的元素个数相等，同时记$G = D/\{v\}$，取A与B中与v相连接的边e_1, e_2，

也即e_1以v_1为起点，以v为终点，e_2以v为起点，以v_2为终点。

并做如下操作：$G = G + v_1v_2$，$A = A/\{e_2\}$，重复以至A为空集。

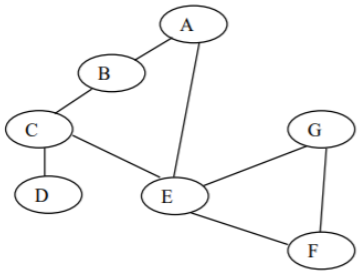
最后得到的G有k个顶点，且每个顶点的度与在G中完全一样。则由归纳假设，G中存在有向的$Euler$环游。

c.任意顶点的度为偶数。

3. 设 G 是具有n个顶点和m条边的无向图，如果 G 是连通的，而且满足$m = n - 1$,证明 G 是树。

若G中有圈，则删去圈上任一条边G仍连通，而每个连通图的边数$e \geq n - 1$，但是删去一条边后$m = n - 2$，这与删去一条边后的G仍是一个连通图矛盾，因此G中无圈，G为树。

5. 下面的无向图以邻接链表存储，而且在关于每个顶点的链表中与该顶点相邻的顶点是按照字母顺序排列的。试以此图为例描述讲义中算法 DFNL 的执行过程。



初始化数组 DFN=0，num=1；

从根节点 A 开始，计算 DFNL(A,null)，

则 DFN(A):=num=1;L(A):=num=1;num:=1+1=2

检查到 A 的邻接点 B，由于 DFN(B)=0，故计算 DFNL(B,A)

DFN(B):=num=2;L(B):=num=2;num:=2+1=3

检查到 B 的邻接点为 A，但 DFN(A)=1，且 A 为 B 的父节点，故跳过

检查到 B 的邻接点 C，DFN(C)=0，计算 DFNL(C,B)

则 DFN(C):=num=3;L(C):=num=3;num:=3+1=4

B 已被检索过，且为 C 的父节点，故跳过；检测到 C 的邻接点 D，计算 DFNL(D,C)

则 DFN(D):=num=4;L(D)=num=4;num:=4+1=5

C 跳过，且 D 的邻接链表均已遍历，返回到节点 C，检测 C 的邻接节点 E，计算计算 DFNL(E,C)

DFN(E):=num=5;L(E):=num=5;num:=5+1=6

检查到 E 的邻接点 A，由于 A 已被检测过且 A 不为 E 的父节点(也即 AE 为余边)，则 L(E):=min(L(E),DFN(A))=1

C 跳过，检测 E 的邻接点 F，计算 DFNL(F,E)

则 DFN(F):=num=6;L(F):=num=6;num:=1+6=7

E 跳过，检测到 F 的邻接点 G，计算 DFNL(G,F)

则 DFN(G):=num=7;L(G):=num=7;num:=1+7=8

由于 E 不是 G 的父节点，且 E 已被检测，因此 L(G)=min(L(G),DFN(E))=5

F 跳过，此时 DFNL(G,F)计算结束，L(F):=min(L(F),L(G))=5

此时 DFNL(F,E)计算结束，L(E)=min(L(E),L(F))=1

此时 DFNL(E,C)计算结束，L(C)=min(L(C),L(E))=1

此时 DFNL(C,B)计算结束，L(B)=min(L(B),L(C))=1

此时 DFNL(B,A)计算结束，L(A)=min(L(A),L(B))=1

6.对图的另一种检索方法是 D-Search。该方法与 BFS 的不同之处在于将队列换成栈，即下一个要检测的节点是最新加到未检测节点表的那个节点。

1）写一个 D-Search 算法；

2）证明由节点 v 开始的 D-Search 能够访问 v 可到达的所有节点；

3）你的算法的时、空复杂度是什么？

1）伪代码：

```
def DBFT(G,v):
    count=0#用来计数到目前为止已经被访问过的节点数
    branch=0#图G中的连通分支数
    for i to n:
        s[i]=0#用来表示各顶点是否曾被检测过，初始化为0。
    for i to v:
        if s[i]==0 :
            DBFS(i)
            branch+=1
def DBFS(v):
#BFS搜索G，从v开始，visited表示各顶点被访问的序号数
#S为栈
    S.push_back(v)
    while S 非空:
        u=S.pull_head()
        count+=1
        visited[u]=count
        for w in u的所有邻接点:
            if s[w]=0:
```

```
S.push_back(w)
s[w]=1
```

2）证明：除了节点 v 之外，只有当节点 w 满足 s[w]=0 才会被压入栈中，因此每个节点至多有一次被压入栈中，不会出现重复检测的现象，而栈中存储了 v 可达的所有节点，在栈非空时算法不会停止，因此可保证从 v 开始，该算法能够访问 v 可达的所有节点

3）对于栈空间，只有当 w 满足 s[w]=0 才会被压入栈中，因此需要的栈空间至多是 n-1，visited 需要的空间为 n，其余变量所用的空间都为 O(1)，因此空间复杂度为$\Theta(n)$

假设采用邻接链表，则 DBFS 中 while 需要做 n 次，for 需要 d(u)次（表示 u 节点的链表深度），又 visited、s、count 的赋值均需要 n 次，故时间复杂度为$\Theta(n+k)$

假设采用邻接矩阵，while 需要做 n 次，for 需要做n^2次，同时 visited、s、count 的赋值均需要 n 次，故时间复杂度为$\Theta(n^2)$