一、基础题

1. 已知一棵树边的集合为{<I,M>, <I,N>, <E,I>, <B,E>,<B,D>,<A,B>,<G,J>,<G,K>,<C,G>,<C,F>,<H,L>,<C,H>,<A,C>}，请画出这棵树，并回答下列问题：
   (1) 哪个是根结点？（A）
   (2) 哪些是叶子结点？（D，M，N，F，J，K，L）
   (3) 哪个是结点 G 的双亲？（C）
   (4) 哪些是结点 G 的祖先？（A,C）
   (5) 哪些是结点 G 的孩子？（J，K）
   (6) 哪些是结点 E 的子孙？（I，M，N）
   (7) 哪些是结点 E 的兄弟？哪些是结点 F 的兄弟？（E 的兄弟是 D，F 的兄弟是 G 和 H）
   (8) 结点 B 和 N 的层次号分别是什么？（2,5）
   (9) 树的深度是多少？（5）
   (10) 以结点 C 为根的子树的深度是多少？（3）

2. 一棵深度为 H 的满 k 叉树有如下性质：第 H 层上的结点都是叶子结点，其余各层上每个结点都有 k 课非空子树。如果按层次顺序从 1 开始对全部结点编号，问：
   (1) 各层的结点数目是多少？（第 i 层的结点数为k^{i-1}）
   (2) 编号为 p 的结点的父亲结点（若存在）的编号是多少？（若 p=1，则该结点为根，没有双亲结点；否则双亲结点的编号为$\left\lfloor \frac{p+(k-2)}{k} \right\rfloor$

   说明：考查第 i 个结点和它的孩子的编号之间的关系。第 i 个结点前有 i-1 个结点，因此，第 i 个结点的孩子的序号从(i-1)*k+1 后开始编号。即，第 i 个结点的孩子的编号是：(i-1)*k+1+1，(i-1)*k+1+2，…，(i-1)*k+1+k。转换为与 i 相关的表示方式为：i*k-(k-2)，i*k-(k-3)，…，i*k，i*k+1，显然这些孩子中任意一个 p 的双亲的编号 i=$\left\lfloor \frac{p+(k-2)}{k} \right\rfloor$
   ）
   (3) 编号为 p 的结点的第 i 个儿子结点（若存在）的编号是多少？

解：由上分析可知，编号为 p 的结点的孩子编号为：p*k-(k-2)，p*k-(k-3)，…，p*k，p*k+1，其中第 i 个的编号为：p*k-(k-2)+(i-1)=(p-1)*k+i+1

   (4) 编号 p 的结点有右兄弟的条件是什么？其右兄弟的编号是多少？

解：由上分析可知，编号为 i 的结点的孩子编号为：i*k-(k-2)，i*k-(k-3)，…，i*k，i*k+1，其中除了编号为 i*k+1 的结点外，其他结点都有右兄弟，所以（p-1）%k 不等于 0 时（即 p 不为 i*k+1 时），都有右兄弟，其右兄弟编号显然为 p+1。

3. 已知一棵度为 k 的树中有 n_1 个度为 1 的结点，n_2 个度为 2 的结点，。。。，n_k 个度为 k 的结点，问该树中有多少个叶子结点？

解：已知结点总数 $n=n_0+n_1+n_2+…+n_k$；从分支角度分析，除了根结点外，每一个结点都与唯一双亲通过分支连接。即，除根结点外，分支数和结点数一一对应。所以，结点总数也可以表示为分支总数+1= $1+n_1+2*n_2+3*n_2+…+k*n_k$

显然：$n_0+n_1+n_2+…+n_k=1+n_1+2*n_2+3*n_2+…+k*n_k$

所以：$n_0 =1+ 1*n_2+2*n_2+…+(k-1)*n_k=1+\sum_{i=2}^{k}(i-1)n_i$

4. 一棵含有 n 个结点的 k 叉树，可能达到的最大和最小深度各为多少？

解：最大深度显然为 n，即“单支”度为 1 的树。最小深度出现在树为完全 k 叉树时。设 n 个结点的 k 叉树深度为 i，则第 1 到第 i-1 层必然形成满 k 叉，因此存在如下关系：

$$\frac{k^{i-1}-1}{k-1} < n \le \frac{k^{i}-1}{k-1}$$

$$k^{i-1} \le n(k-1) < k^{i}$$

$$i-1 \le \log_k n(k-1) < \mathrm{i}$$

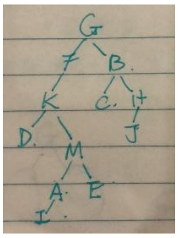
所以，最小深度为$\lfloor \log_k n(k-1) \rfloor + 1$

5. 画出和下列已知序列对应的二叉树 T：

树的先根次序访问序列为：GFKDMAIEBCHJ;

树的后根次序访问序列为：DIAEMKFCJHBG。

解：已知二叉树的先序遍历顺序为 DLR，而后序为 LRD，因此，可以通过逐层比较分析得到字母（集）相对不变的是左右子树，变化了的是根结点。这样逐层递归分析可得二叉树为：



二、算法题

1. 假定用两个一维数组 L[n+1]和 R[n+1]作为有 n 个结点的二叉树的存储结构，L[i]和 R[i]分别指示结点 i(i=1,2,…,n)的左孩子和右孩子，0 表示空。试写一个算法判别结点 u 是否结点 v 的子孙。

解：本题考查大家在顺序存储结构上编写二叉树遍历算法的能力。遍历结点 v 的左右子树，若结点 u 是左或右子树中的结点，则说明 u 是 v 的子孙。

算法需要不断递归遍历结点的孩子，因此存储结构 L 和 R 应保存各结点左右孩子的序号。

参考代码为：`IsDescendents`

2. 假设在二叉链表的节点中增设两个域：双亲域（parent）以指示其双亲结点；标志域（mark 取值 0..2）以区分在遍历过程中到达该结点时应继续向左或向右或访问该结点。试以此存储结构编写不用栈进行后序遍历的递推形式的算法。

解：本题考查对遍历算法的理解和运用。后序遍历算法会三次到达一个结点。第一次到结点时，应该遍历其左子树；第二次到达结点时，即对其左子树完成遍历时，应遍历其右子树；第三次到达结点时，即对其右子树完成遍历时，应访问该结点，而后回到其双亲结点，按同样规则继续处理，直到所有结点都被遍历。

可见，只要设置标识记录到达结点的次数（本例中，利用 mark）就能完成上述处理，当然返回双亲所需的双亲指针也是必不可少的。需要注意的时，按后序遍历规则，只有访问了一个结点（第 3 次到达该结点）时，已访问结点计数器才能自增 1，参考代码详见 `TraverseWithoutStackOrRecursion`。

3. 编写递归算法，将二叉树中所有结点的左右子树相互交换。

解：本题考查对递归机制的理解和运用。按递归的方式分析，题目所求任务分为三个步骤，分别是:交换**左子树**中结点的左右子树,交换**右子树**中结点的左右子树,交换**结点**左右子树。据此很容易写出递归算法。本题按照后序遍历完成任务,按照其他遍历顺序完成任务也一样。参考代码详见 `SwapTree`。

4. 编写按层次顺序（同一层自左向右）遍历二叉树的算法。

解：本题考查对数据结构的分析。按照层次遍历的要求，同层越靠左的结点，其孩子在下一层的遍历中，越应被优先访问。也就是说，访问顺序应该是“先进先出”。所以可以选用队列作为辅助数据结构，保存待访问的结点序列。参考代码详见 `TraverseWithQueue`。