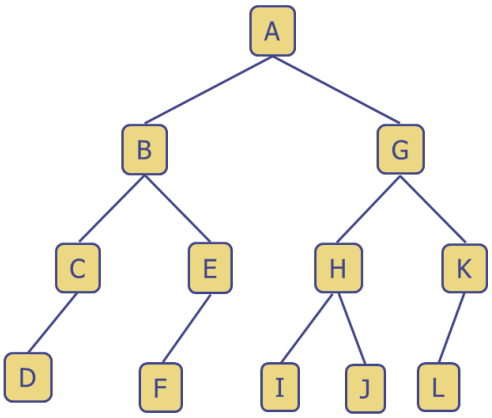
Tutorial 5

Question 1: Trees

For each of the following trees, fill in its corresponding three tables:

- The first table is about generic properties of the tree.
- The second table is about properties for specific nodes in the tree.
- The third table contains an array. For each node in the tree, you should mark which position in the array it should occupy.

Tree 1



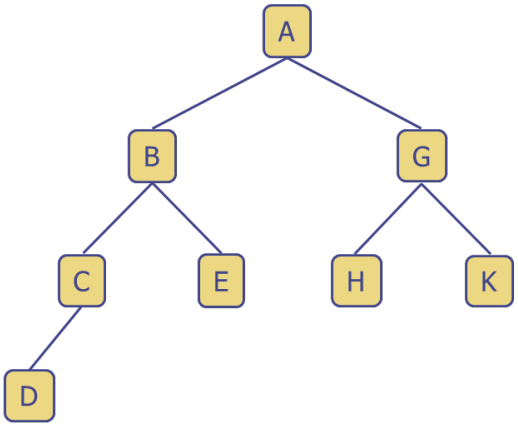
Property	Answer
Root Node	A
Internal Nodes	A, B, C, E, G, H, K
External Nodes	D, F, I, J, L
Height of tree	3
Inorder traversal	DCBFEAIHJGLK
Preorder traversal	ABCDEFGHIJKL
Postorder traversal	DCFEBIJHLKGA

	Depth	**Ancestor**	**Descendants**

A	**0**	**A**	**A, B, G, C, E, H, K, D, F, I, J, L**
B	**1**	**B, A**	**B, C, D, E, F**
C	**2**	**C, B, A**	**C, D**
D	**3**	**D, C, B, A**	**D**
E	**2**	**E, B, A**	**E, F**
F	**3**	**F, E, B, A**	**F**
G	**1**	**G, A**	**G, H, K, I, J, L**
H	**2**	**H, G, A**	**H, I, J**
I	**3**	**I, H, G, A**	**I**
J	**3**	**J, H, G, A**	**J**
K	**2**	**K, G, A**	**K, L**
L	**3**	**L, K, G, A**	**L**

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
-	**A**	**B**	**G**	**C**	**E**	**H**	**K**	**D**		**F**		**I**	**J**	**L**	

Tree 2



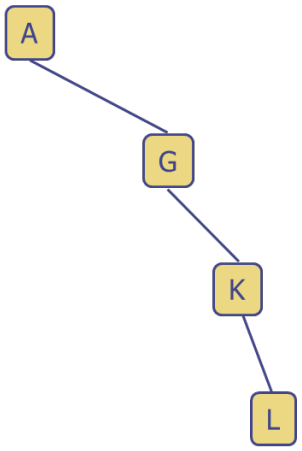
Property	Answer
Root Node	A
Internal Nodes	A, B, C, G

External Nodes	D, E, H, K
Height of tree	3
Inorder traversal	DCBEAHGK
Preorder traversal	ABCDEGHK
Postorder traversal	DCEBHKGA

	Depth	**Ancestor**	**Descendants**
A	**0**	**A**	**A, B, G, C, E, H, K, D**
B	**1**	**B, A**	**B, C, E, D**
C	**2**	**C, B, A**	**C, D**
D	**3**	**D, C, B, A**	**D**
E	**2**	**E, B, A**	**E**
G	**1**	**G, A**	**G, H, K**
H	**2**	**H, G, A**	**H**
K	**2**	**K, G, A**	**K**

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
-	**A**	**B**	**G**	**C**	**E**	**H**	**K**	**D**							

Tree 3

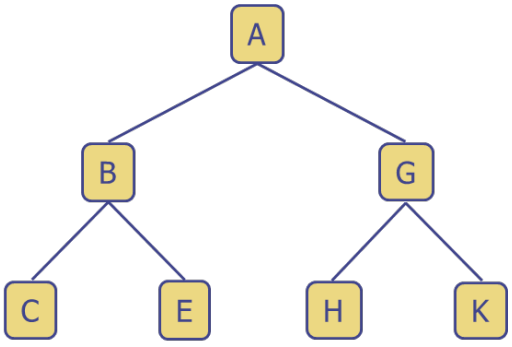


Property	Answer
Root Node	A
Internal Nodes	A, G, K
External Nodes	L
Height of tree	3
Inorder traversal	AGKL
Preorder traversal	AGKL
Postorder traversal	LKGA

	Depth	**Ancestor**	**Descendants**
A	**0**	**A**	**A, G, K, L**
G	**1**	**G, A**	**G, K, L**
K	**2**	**K, G, A**	**K, L**
L	**3**	**L, K, G, A**	**L**

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
-	**A**		**G**				**K**								**L**

Tree 4



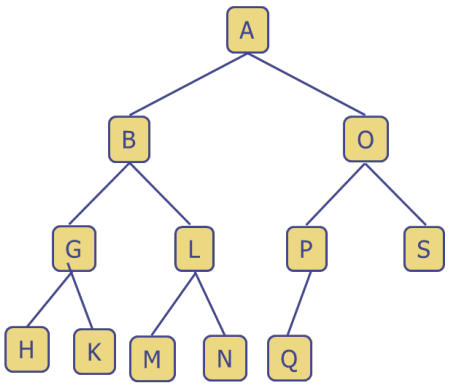
Property	Answer

Root Node	A
Internal Nodes	A, B, G
External Nodes	C, E, H, K
Height of tree	2
Inorder traversal	CBEAHGK
Preorder traversal	ABCEGHK
Postorder traversal	CEBHKGA

	Depth	**Ancestor**	**Descendants**
A	**0**	**A**	**A, B, G, C, E, H, K**
B	**1**	**B, A**	**B, C, E**
C	**2**	**C, B, A**	**C**
E	**2**	**E, B, A**	**E**
G	**1**	**G, A**	**G, H, K**
H	**2**	**H, G, A**	**H**
K	**2**	**K, G, A**	**K**

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
-	**A**	**B**	**G**	**C**	**E**	**H**	**K**

Tree 5



Property	Answer
Root Node	A
Internal Nodes	A, B, O, G, L, P
External Nodes	H, K, M, N, Q, S
Height of tree	3
Inorder traversal	HGKBMLNAQPOS
Preorder traversal	ABGHKLMNOPQS
Postorder traversal	HKGMNLBQPSOA

	Depth	**Ancestor**	**Descendants**
A	**0**	**A**	**A, B, O, G, L, P, S, H, K, M, N, Q**
B	**1**	**B, A**	**B, G, L, H, K, M, N**
G	**2**	**G, B, A**	**G, H, K**
H	**3**	**H, G, B, A**	**H**
K	**3**	**K, G, B, A**	**K**
L	**2**	**L, B, A**	**L, M, N**
M	**3**	**M, L, B, A**	**M**
N	**3**	**N, L, B, A**	**N**
O	**1**	**O, A**	**O, P, S, Q**
P	**2**	**P, O, A**	**P, Q**
Q	**3**	**Q, P, O, A**	**Q**
S	**2**	**S, O, A**	**S**

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
-	**A**	**B**	**O**	**G**	**L**	**P**	**S**	**H**	**K**	**M**	**N**	**Q**			

Additional questions:

Proper binary tree（又称strict binary tree）：每个内部节点**要么有两个子节点**，要么**没有子节点**（即叶节点）。

1. Which of the above trees, if any, are proper binary trees? Tree 4
2. How big of an array do we need to store an arbitrary binary tree of height h?

如果你要用数组表示一个任意高度为 h 的二叉树，**就需要数组长度至少为** 2^{h+1} - 1 才能保证不溢出。

这个公式是完全二叉树的节点数量公式。

$$1 + 2 + \ldots + 2^h = 2^{h+1} - 1$$

3. We have shown how to use an array representation for binary trees. How would we extend this to work on ternary trees?
   Store the root at index 1.
   For every internal node at index n, store its first child at index 3n-1, its second child at 3n, and its third child at 3n+1.

✅ **二叉树的数组表示回顾（1-based indexing）：**

- 根节点在 index = 1
- 对任意节点 index = n：
  - 左子：$2n$
  - 右子：$2n + 1$

✅ **扩展到三叉树（Ternary Tree）：**

你的写法：

> For every internal node at index n,
>
> - First child at $3n - 1$
> - Second child at $3n$
> - Third child at $3n + 1$