Lab 02

What is the minimum possible depth of a binary tree with 300 leaves?

The shallowest tree is the balanced ("fullest") tree. (Adding nodes to balance an unbalanced tree either leaves the same or increases the number of leaves.)

In a balanced tree of depth d, we only need to consider the capacity of the deepest layer d, which is 2^d. (When we add a node to a balanced tree, we can primarily append to a node in depth $d-1$. If we cannot fit 300 nodes in the "previous layer", we need to start filling up layer d.)

We seek the smallest layer that could fit at least 300 nodes. $2^d \geq 300$. Thus, $d = \lceil \log_2(300) \rceil$.

What is the minimum possible depth of a ternary tree with 300 leaves?

Same as before, only with $\log_3$. $d = \log_3(300)$

There are n nodes in a given regular binary tree. What is the number of its leaves?

We must derive a rule for a regular binary tree's number of leaves. Recall that any node in a regular binary tree has either zero or exactly two child nodes. Now start with a tree with only a root node. Having no child nodes, it is clearly a regular tree. It has one leaf, the root node itself. Create the next smallest tree by adding two child nodes to the root. Both new nodes have no child nodes, thus we have a regular tree. With this addition, we reduced the number of leaves by 1 (root is no longer a leaf), but we gained 2 new leaves. Now we may infer that for every two nodes, we gain one leaf. After all, $l = \frac{n+1}{2}$. Note that the expression always yields an integer, because all regular binary trees always have an odd number of nodes.

A given binary tree T has exactly three leaves. Therefore

1. T has at most two inner (non-leaf) nodes,
2. number of inner nodes is not limited,
3. all leaves are in the same depth,
4. all leaves cannot have the same depth,
5. T is regular.

The question is about whether the *implications* hold.

1. Implication does not hold. Imagine a tree with a single arbitrarily long string of nodes. One leaf is at the end, and the other two are placed arbitrarily. Most nodes have only one child, which is perfectly acceptable in binary trees. (Would not be possible in regular binary trees, though.) For example, we have found an arbitrarily deep binary tree with exactly three leaves.
2. Yes. See above.
3. Possibly, but not necessarily. See above.
4. No. A balanced binary tree with six nodes is a counterexample.
5. No. See above.

Algorithm A traverses a balanced binary tree with n nodes. The algorithm performs an additional procedure in each node whose complexity is $\Theta(n^2)$. What is the asymptotic complexity of A?

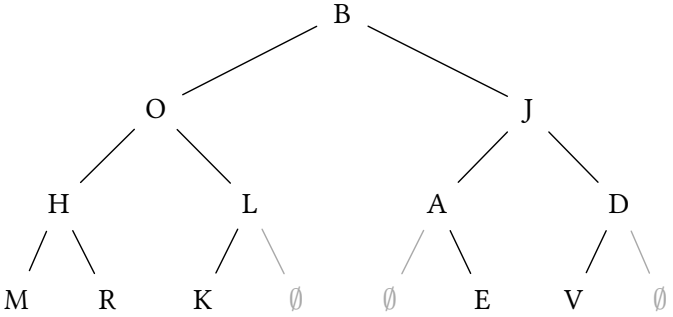
We need $\Theta(n)$ operations to traverse all nodes. (Assuming a doubly-linked binary tree, we must traverse each edge twice. We also know that any tree with n nodes has $n-1$ edges.) I.e., the total complexity is $\Theta(n^3)$.

Algorithm A traverses a binary tree with depth n. The total number of operations performed in depth k equals $k + n$. Each operation has a constant asymptotic complexity. What is the asymptotic complexity of A?

Luckily, we do not need to bother with the number of nodes - the complexity concerns only the depth for some reason. (Who knows what the algorithm might be doing? That is not the point of this exercise.) Thus, for layer 0 we have $n + 0$ operations, for layer 1 we have $n + 1$ operations, for layer 2 there are $n + 2$ operations, etc, up to layer n with $n + n$ operations. The answer is thus

$n \cdot n + \sum_{i=0}^{n} i = n^2 + \frac{n(1+n)}{2} \in \Theta(n^2)$

Algorithm A traverses the given tree, and in each node, it prints the character stored in that node. Write the output of the algorithm when the traversal is 1. inorder, 2. preorder, 3. postorder.

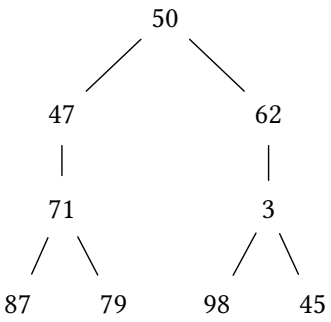


1. inorder: mhroklbeajvd
2. preorder: bohmrlkjaedv
3. postorder: mrhkloeavdjb

A method first prints the keys of the nodes of a binary tree using the Inorder traversal, and then it prints the keys of the nodes using the Preorder traversal.

The result is Inorder: 45 71 98 47 50 62 87 3 79 Preorder: 50 47 71 45 98 62 3 87 79

a) Reconstruct the tree completely.



b) Write an algorithm that will reconstruct any binary tree when given a sequence S1 of keys of the nodes produced by the Inorder traversal and a sequence S2 of keys of the nodes produced by the Preorder traversal.

Use the “preorder” to pop local roots; search popped values in “in-order”, to determine whether to attach intermediate nodes as left or right sub-trees. Recurse into the sub-tree.

Suggest a method for a given value n to generate a tree with n nodes whose depth will belong to $\Theta(n)$.

First, recall that n nodes can fit into a tree of depth $\log(n)$. Then realize that $\forall n > 1 : n > \sqrt{n} > \log(n)$. Thus, we are required to make a tree deeper than necessary while certainly having enough nodes. The algorithm can thus be designed to primarily build a sparse (single-branch) tree of depth $\lceil\sqrt{n}\rceil$. When depth is satisfied, excess nodes are appended arbitrarily to the shallower nodes until the deepest node needs to be extended to comply with the depth requirement.

We must traverse a regular binary tree and visit all its n nodes. We can move along each edge only in the direction from the root towards a leaf. We can also jump from any node directly back to the root. Each move along one edge and each jump to the root takes one millisecond. Compute the best possible complexity of the traversal. The tree is

A) perfectly balanced: The depth is thus $\lfloor\log(n)\rfloor$. Capacity of the last layer (maximum number of nodes) is $\lceil\log(n)\rceil$. We need to visit all leaves, and to visit any leaf, we need to traverse all parent nodes. I.e., the result is $\Theta(\lfloor\log_2(n)\rfloor \cdot \lceil\log_2(n)\rceil) = \Theta(\log^2(n))$

B) maximally disbalanced (its depth is (n−1)/2): Leaves are in depths of $1, 2, 3, \ldots, \frac{n-1}{2} - 1, \frac{n-1}{2}, \frac{n-1}{2}$, which also correspond to the number of operations to visit respective leaves. Summing the series yields $\frac{\frac{n-1}{2}\cdot(\frac{n-1}{2}+1)}{2}$ which is $\Theta(n^2)$

An arithmetic expression containing only positive integers, brackets and symbols of operations $+, -, \times, /$ can be represented as a binary tree.

Example: The expression $6 + (4 - 3 + 5) \times (9 - 7)$ is represented by the tree in the picture. Write a node representation and a function whose input is the reference to the tree root, and the return value is equal to the value of the expression the tree represents.