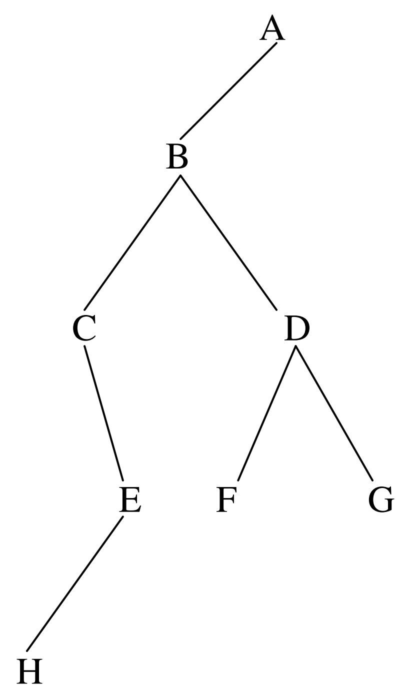
CONCEPT EXERCISES

9.1 Answer the questions below about the following binary tree:



a. What is the root element?
A

b. How many elements are in the tree?
8

c. How many leaves are in the tree?
3

d. What is the height of the tree?
4

e. What is the height of the left subtree?
3

f. What is the height of the right subtree?
–1

g. What is the level of F?
3

h. What is the depth of C?
2

i. How many children does C have?
1

j. What is the parent of F?
D

k. What are the descendants of B?
C, D, E, F, G, and H

l. What are the ancestors of F?
A, B, and D

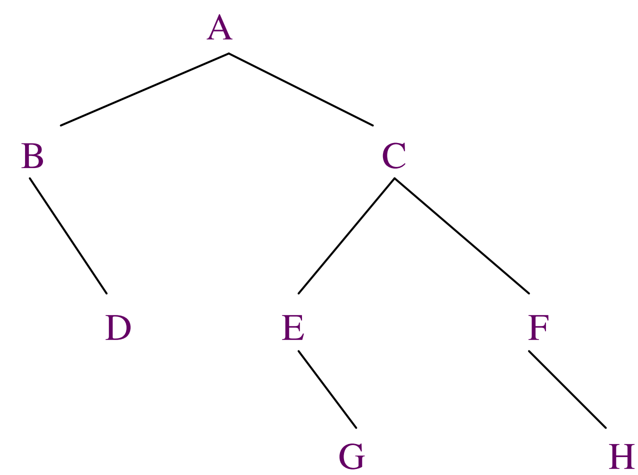
m. What would the output be if the elements were written out during an inOrder traversal?
C, H, E, B, F, D, G, A

n. What would the output be if the elements were written out during a postOrder traversal?
H, E, C, F, G, D, B, A

o. What would the output be if the elements were written out during a preOrder traversal?
A, B, C, E, H, D, F, G

p. What would the output be if the elements were written out during a breadth-first traversal?
A, B, C, D, E, F, G, H

9.2 **a.** Construct a binary tree of height 3 that has 8 elements.



b. Can you construct a binary tree of height 2 that has 8 elements?

No. The maximum number of elements, n, in a binary tree of height 2 is attained if the tree is full, and then

$(n + 1) / 2 = 2^2$ by the Binary Tree Theorem

For a full binary tree of height 2, $n = 7$.

c. For n going from 1 to 20, determine the minimum height possible for a binary tree with n elements.

n	minimum height
1	0
2	1
3	1
4	2
5	2
6	2
7	2
8	3
9	3
10	3
11	3
12	3
13	3

14	3
15	3
16	4
17	4
18	4
19	4
20	4

d. Based on your calculations in part c, try to develop a formula for the minimum height possible for a binary tree with *n* elements, where *n* can be any positive integer.

height(t) >= floor($\log_2 n$)

e. Use the Principle of Mathematical Induction (Strong Form) to prove the correctness of your formula in part d.

For n = 1, 2, …, let S_n be the statement

If *t* is a binary tree with *n* elements, height(*t*) >= floor ($\log_2 n$).

(1) Base case:

Let *t* be a binary tree with 1 element. Then the height of the tree must be 0. That is,

height(*t*) = 0 = floor ($\log_2 1$)

So S_1 is true.

(2) Inductive case:

Let *n* be any positive integer, and assume S_1, S_2, … , S_n are true. We need to show that S_{n+1} is true. Let *t* be a binary tree with *n* + 1 elements. There are two cases to consider:

a. *n* + 1 is even. Either leftTree(*t*) or rightTree(*t*) has at least (*n* + 1) / 2 elements. For specificity, assume leftTree(t) has *n1* elements, where *n1* >= (*n* + 1) / 2. Since 1 <= *n1* <= *n*, the Induction Hypothesis applies to leftTree(*t*). That is,

height(leftTree(*t*)) >= floor($\log_2(n1)$)

Since floor and $\log_2$ are both non-decreasing functions and $n1 >= (n + 1) / 2$,

$$\text{floor}(\log_2(n1)) >= \text{floor}(\log_2((n + 1) / 2))$$

$$= \text{floor}(\log_2(n + 1) - \log_2(2))$$

$$= \text{floor}(\log_2(n + 1)) - 1$$

So $\text{height}(t) >= 1 + \text{height}(\text{leftTree}(t))$

$$>= 1 + \text{floor}(\log_2(n + 1)) - 1$$

$$= \text{floor}(\log_2(n + 1))$$

We conclude that S_{n+1} is true if $n + 1$ is even.

b. $n + 1$ is odd. Either leftTree(t) or rightTree(t) has at least $n / 2$ elements. For specificity, assume leftTree(t) has $n1$ elements, where $n1 >= n / 2$. Since $1 <= n1 <= n$, the Induction Hypothesis applies to leftTree(t). That is,

$$\text{height}(\text{leftTree}(t)) >= \text{floor}(\log_2(n1))$$

Since floor and $\log_2$ are both non-decreasing functions and $n1 >= n / 2$,

$$\text{floor}(\log_2(n1)) >= \text{floor}(\log_2(n / 2))$$

$$= \text{floor}(\log_2 n - \log_2(2))$$

$$= \text{floor}(\log_2 n) - 1$$

$$= \text{floor}(\log_2(n + 1)) - 1$$

(because $n + 1$ is odd)

So $\text{height}(t) >= 1 + \text{height}(\text{leftTree}(t))$

$>= 1 + \text{floor}(\log_2(n + 1)) - 1$

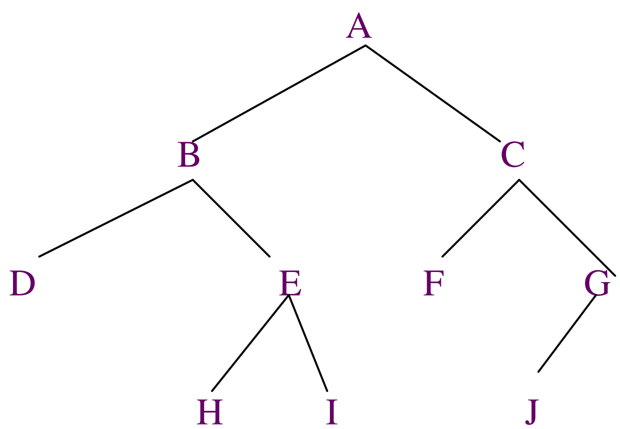
$= \text{floor}(\log_2(n + 1))$

We conclude that S_{n+1} is true if $n + 1$ is odd.

Therefore, by the Principle of Mathematical Induction (Strong Form), S_n is true for all positive integers n.

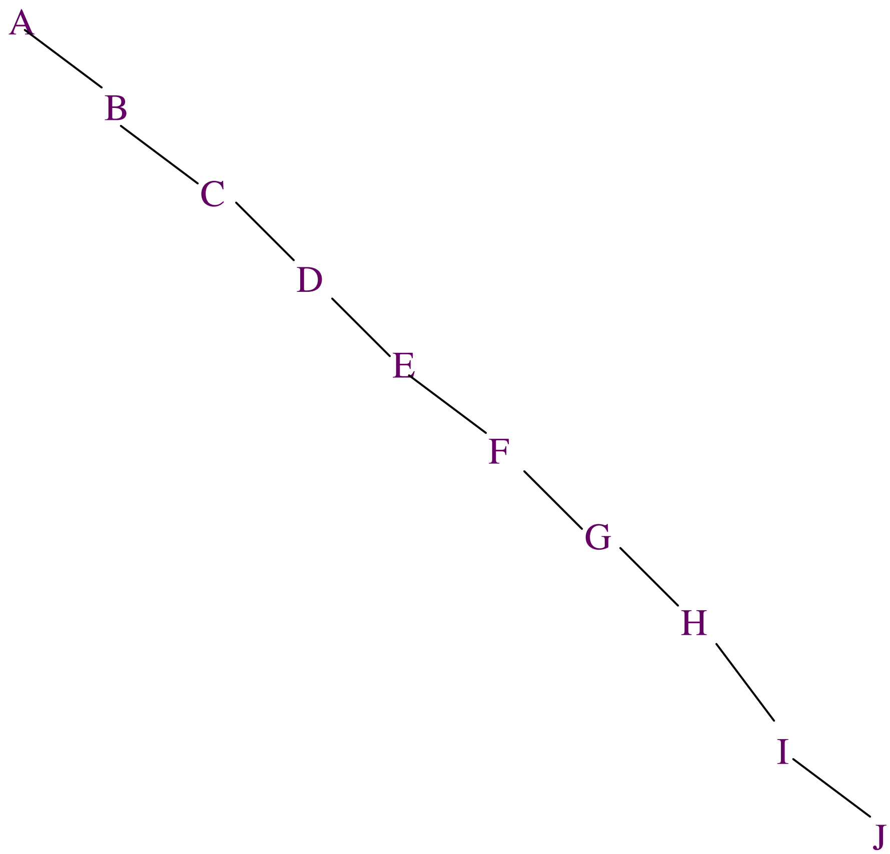
9.3 **a.** What is the maximum number of leaves possible in a binary tree with 10 elements? Construct such a tree.

5

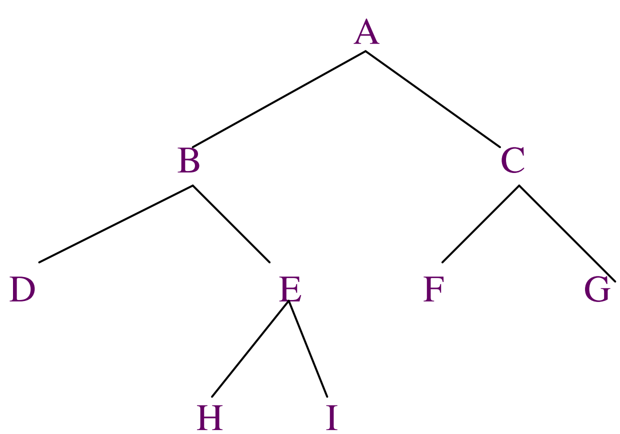


b. What is the minimum number of leaves possible in a binary tree with 10 elements? Construct such a tree.

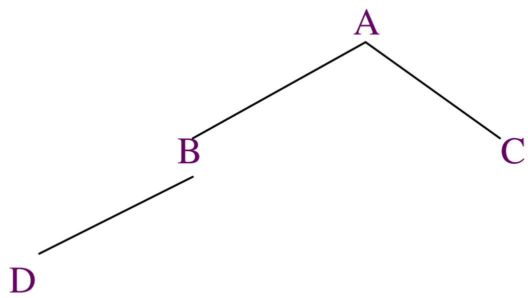
1



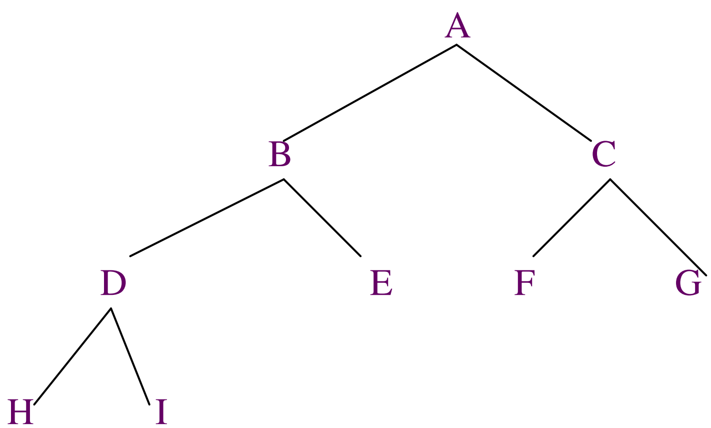
9.4 **a.** Construct a two-tree that is not complete.



b. Construct a complete tree that is not a two-tree.



c. Construct a complete two-tree that is not full.



d. How many leaves are there in a two-tree with 17 elements?

9

e. How many leaves are there in a two-tree with 731 elements?

366

f. A non-empty two-tree must always have an odd number of elements. Why? Hint: Use the Binary Tree Theorem and the fact that the number of leaves must be an integer.

By part 3 of the binary tree theorem, if t is a two tree,

leaves(t) = (n(t) + 1) / 2.0

Therefore,

2.0 leaves(t) – 1 = n(t), which implies that n(t) must be odd.

g. How many elements are there in a full binary tree of height 4?

31

h. How many elements are there in a full binary tree of height 12?

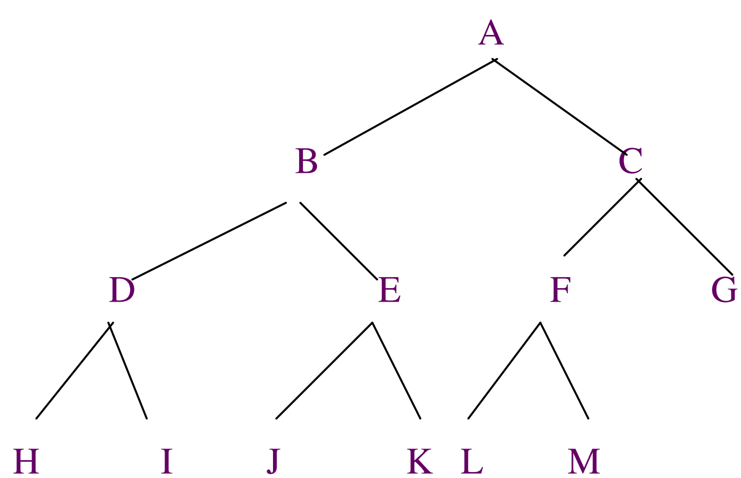
8191

i. Use the Binary Tree Theorem to determine the number of leaves in a full binary tree with 63 elements.

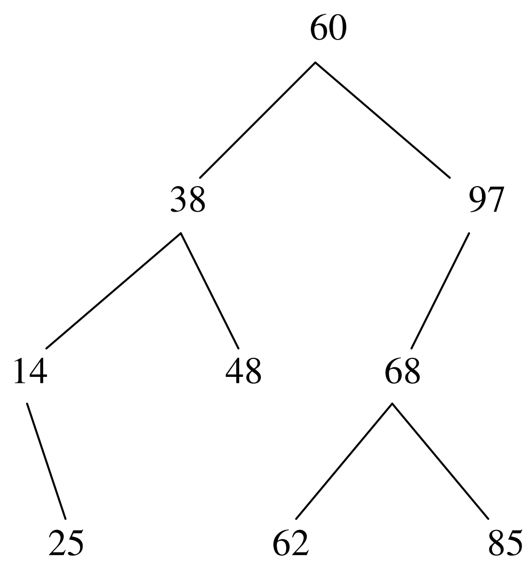
Because a full binary tree must also be a two tree,

leaves(t) = (n(t) + 1) / 2.0 = (63 + 1) / 2.0 = 32

k. Construct a complete two-tree that is not full, but in which the heights of the left and right subtrees are equal.



9.5 For the following binary tree, show the order in which elements would be visited for an inOrder, postOrder, preOrder, and breadthFirst traversal.



inOrder: 14, 25, 38, 48, 60, 62, 68, 85, 97

postOrder: 25, 14, 48, 38, 62, 85, 68, 97, 60

preOrder: 60, 38, 14, 25, 48, 97, 68, 62, 85

breadthFirst: 60, 38, 97, 14, 48, 68, 25, 62, 85

9.6 Show that a binary tree with n elements has $2n + 1$ subtrees (including the entire tree). How many of these subtrees are empty?

For each of the n elements, there are two subtrees, so this accounts for $2n$ subtrees. The entire tree is also a subtree, so the total number of subtrees is $2n + 1$. Each of the n elements is the root of a non-empty subtree, so there are n non-empty subtrees and therefore $n + 1$ empty subtrees.

9.7 Show that if t is a non-empty, complete binary tree, then

$$height(t) = floor(log_2(n(t)))$$

Hint: Let t be a complete binary tree of height k >= 0, and let $t1$ be a full binary tree of height k – 1. Then $n(t1) + 1 <= n(t)$. Use Part 4 of the Binary Tree Theorem to show that $floor(log_2(n(t1) + 1)) = k$, and use Part 1 of the Binary Tree Theorem to show that $floor (log_2(n(t))) < k + 1$.

Because a complete binary tree of height k is full through level $k - 1$, n($t1$) + 1 <= n(t). Both the logarithm and floor functions are non-decreasing, so we get

(a) $floor(log_2(n(t1) + 1)) <= floor(log_2(n(t)))$

Because $t1$ is a full binary tree, we know from Part 4 of the Binary Tree Theorem that

$$(n(t1) + 1) / 2.0 = 2^{height\,(t1)} = 2^{k-1}$$

This implies that n($t1$) + 1 = 2^k, and so

(b) $floor(log_2(n\,(t1) + 1)) = k.$

By Part 1 of the Binary Tree Theorem,

$$(n(t) + 1) / 2.0 <= 2^{height\,(t)} = 2^k$$

This implies that n(t) + 1 <= 2^{k+1}, and so n(t) + 1 < 2^{k+1}, and therefore

$$log_2(n(t)) < k + 1,$$

from which we obtain

(c) $floor(log_2(n(t))) < k + 1$

Combining the results from b, a, and c, we get

$$k = floor(log_2(n\,(t1) + 1)) <= floor(log_2(n(t))) < k + 1$$

All of these expressions are integers, and so

$$floor(log_2(n(t))) = k$$

9.11 Show that in any complete binary tree t, at least half of the elements are leaves.

Hint: if t is empty, there are no elements, so the claim is vacuously true. If the leaf at the highest index is a right child, then t is a two-tree, and the claim follows from part 3 of the Binary Tree Theorem. Otherwise, t was formed by adding a left child to *the* complete two-tree with $n(t) - 1$ elements.

After the hint, all that remains is to prove the claim if t is formed by adding a left child to the complete two-tree with $n(t) - 1$ elements. That two-tree must have had

$$\frac{n(t) - 1 + 1}{2.0}$$

leaves. But t has the same number of leaves because adding a child to an element in a two-tree does not change the number of leaves. So t has $n(t) / 2.0$ leaves: exactly half the number of elements (t must have an even number of elements because a two-tree has an odd number of elements).