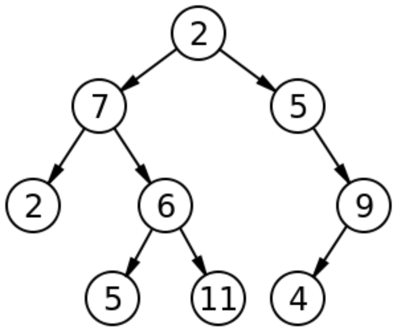
CHAPTER 6 – PROBLEMS

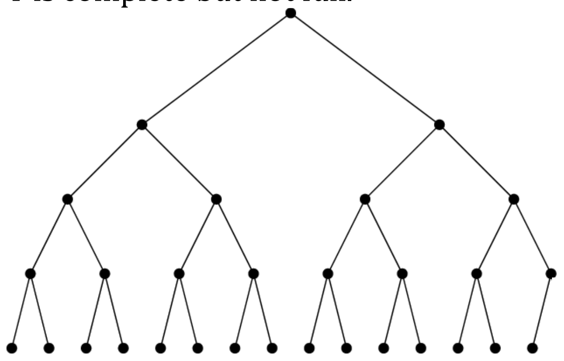
Problem 1. See the binary tree T below. Answer the questions:



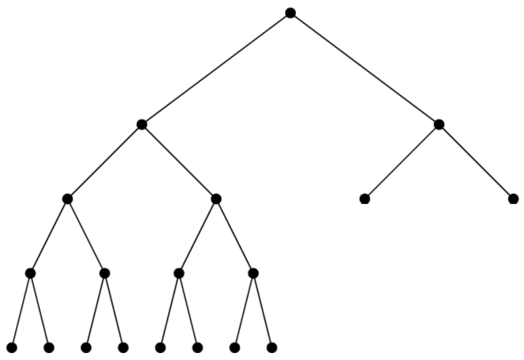
a) Is T a complete binary tree? Is T a full binary tree?
   ➔ No
b) What is the degree of T?
   ➔ 2
c) How many leaves are there? Write down the leaf nodes.
   ➔ 4 leaves: 2, 5, 11, 4
d) How many internal nodes are there? Write down the internal nodes.
   ➔ 5 internal nodes: 2, 7, 5, 6, 9
e) Is T a binary search tree? Why
   ➔ No since node 6 is smaller than 7, so it cannot be at the right subtree of 7.
f) What is the height of T?
   ➔ 3
g) Write down the nodes at level 1, level 2.
   ➔ Level 1: 7, 5
   ➔ Level 2: 2, 6, 9
h) Write down the path from root node to node 11.
   ➔ 2➔7➔6➔11

Problem 2. Draw a binary tree T of height = 4 in these cases:

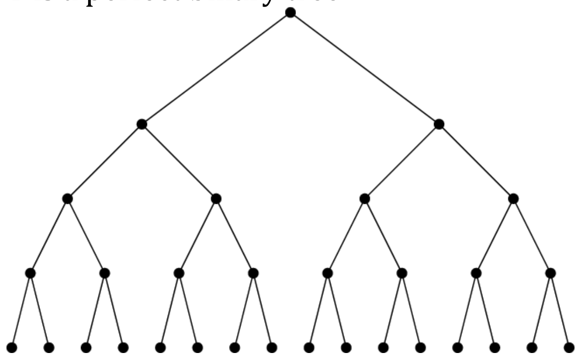
a) T is complete but not full.



b) T is full but not complete.



c) T is a perfect binary tree.

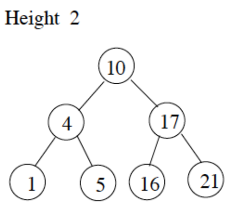


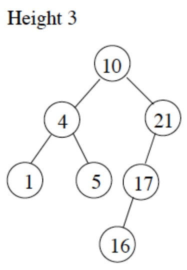
Problem 3. Given a full binary tree T. Let N is the total number of nodes in T, L is the total leaves and I is the total internal nodes + root node. Show that:

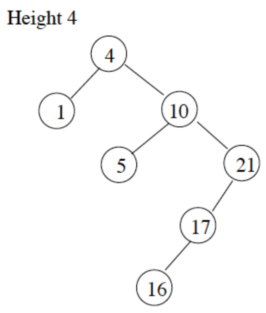
a) L = I + 1
b) N = 2I + 1
c) I = (N – 1)/2
d) L = (N + 1)/2
e) N = 2L – 1
f) I = L – 1

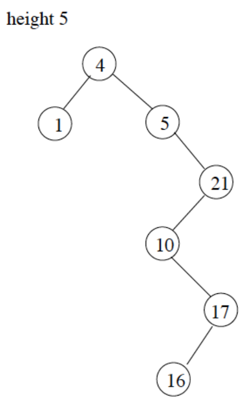
➔ Hint: use induction

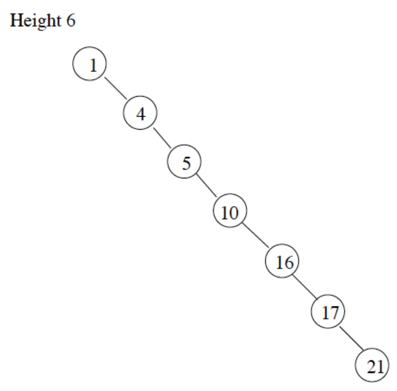
Problem 4. For the set of keys {1, 4, 5, 10, 16, 17, 21}, draw binary search trees of height 2, 3, 4, 5, and 6.











Problem 5. What is the difference between the binary-search-tree property and the min-heap property? Can the min-heap property be used to print out the keys of an n-node tree in sorted order in O(n) time? Explain how or why not.

In a heap, a node.s key is ≥ both of its children.s keys. In a binary search tree, a node.s key is ≥ its left child.s key, but ≤ its right child.s key.
The heap property, unlike the binary-search-tree property, doesn't help print the nodes in sorted order because it doesn't tell which subtree of a node contains the element to print before that node. In a heap, the largest element smaller than the node could be in either subtree.
Note that if the heap property could be used to print the keys in sorted order in O(n) time, we would have an O(n)-time algorithm for sorting, because building the heap takes only O(n) time. But we know (Chapter 2) that a comparison sort must take $\Omega(n \log_2 n)$ time.

Problem 6. Suppose that we have numbers between 1 and 1000 in a binary search tree and want to search for the number 363. Which of the following sequences could not be the sequence of nodes examined?

a) 2, 252, 401, 398, 330, 344, 397, 363.
b) 924, 220, 911, 244, 898, 258, 362, 363.
c) 925, 202, 911, 240, 912, 245, 363.
d) 2, 399, 387, 219, 266, 382, 381, 278, 363.
e) 935, 278, 347, 621, 299, 392, 358, 363.

➔ c, e

Problem 7. Write recursive versions of the functions TREE-MINIMUM and TREE-MAXIMUM in the slide.

```
TREE-MINIMUM($x$)
1.  if $x.left$ == NIL
2.      return $x$
3.  return TREE-MINIMUM($x.left$)

TREE-MAXIMUM($x$)
1.  if $x.right$ == NIL
2.      return $x$
3.  return TREE-MAXIMUM($x.right$)
```

Problem 8. Professor Bunyan thinks he has discovered a remarkable property of binary search trees. Suppose that the search for key k in a binary search tree ends up in a leaf. Consider three sets: A, the keys to the left of the search path; B, the keys on the search path; and C, the keys to the right of the search path. Professor Bunyan claims that any three keys $a \in A, b \in B, c \in C$ must satisfy $a \leq b \leq c$. Give a smallest possible counterexample to the professor's claim.

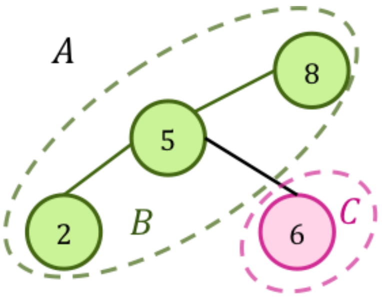
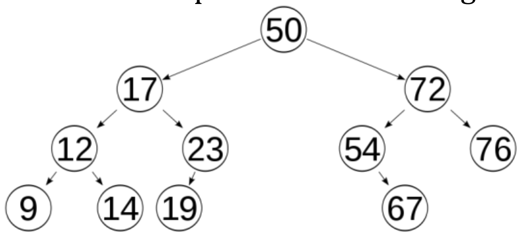


Fig. 1 Counter example to Prof. Bunyan's claim.

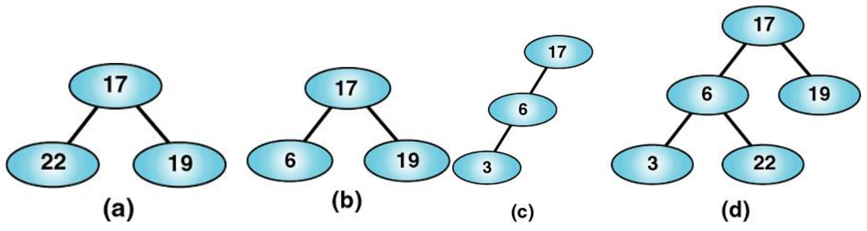
The claim is wrong. A simple counter example is shown in figure 1. In the figure, the search is being done for leaf node 2, so the set B = {8, 5, 2}. There is nothing to the left of the path and so set A = {} . Set C is all elements to the right of the path, so set C = {6} . For any element $a \in A$, and $b \in B$ the claim is true, since A is an empty set. But if set $b = 8$ and $c = 6$, the claim fails to hold.

Problem 9. Write down the nodes on the path of the following tree walks



a) Pre-order tree walk: 50, 17, 12, 9, 14, 23, 19, 72, 54, 67, 76
b) In-order tree walk: 9, 12, 14, 17, 19, 23, 50, 54, 67, 72, 76
c) Post-order tree walk: 9, 14, 12, 19, 23, 17, 67, 54, 76, 72

Problem 10. Which of the following trees are binary search tree:



➔ b, c

Problem 11. Build a binary search tree with the keys: (draw the tree after each insert/delete step)

a) 8, 3, 5, 2, 20, 11, 30, 9, 18, 4. Delete the nodes: 5, 20.
b) 5, 7, 10, 12, 9, 8, 3, 1, 4. Delete the nodes 3, 10, 5.

Problem 12. Consider the following sequence of operations on an initially empty search tree:

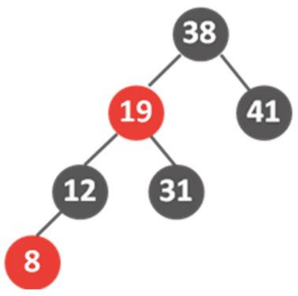
Insert 10
Insert 100
Insert 30
Insert 80
Insert 50
Remove 10
Insert 60
Insert 70
Insert 40
Remove 80
Insert 90
Insert 20
Remove 30
Remove 70

What does the tree look like after these operations execute if the tree is:

i) A binary search tree?
j) An AVL tree?
k) A Red-black tree?
l) A 2-3 tree?
m) A 2-3-4 tree?

Problem 13. Draw a Red-Black Tree that is not an AVL tree structure. Explain your answer.

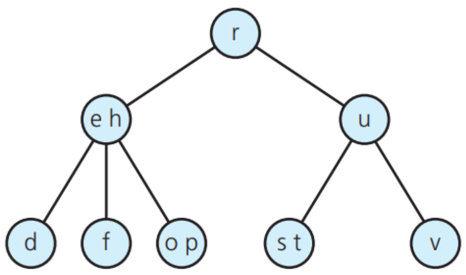
➔ The following tree is not an AVL tree since the different height of the left subtree and the right subtree of the root node is 2 (larger than 1)



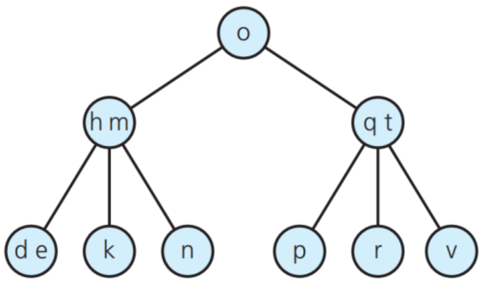
Problem 14. Is it possible to have all black nodes in a Red-Black tree? Give an example for your answer.

Yes, a tree with all nodes black is a red-black tree. Moreover, the tree is also a perfect binary tree (all leaves are at the same depth or same level, and in which every parent has two children).
And it's the only tree whose Black height equals to its tree height.

Problem 15. Given the following 2-3 tree. Draw the tree that results after inserting *k, b, c, y,* and *w* into the tree.



Problem 16. Given the following 2-3 tree. Draw the tree that results after removing *t, e, k,* and *d* from the tree.

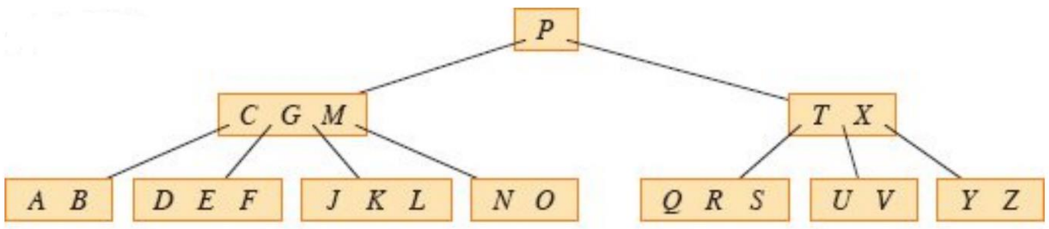


Problem 17. Draw the 2-3-4 tree that results from inserting *o, d, j, h, s, g,* and *a*, in the order given, into a 2-3-4 tree that contains a single node whose value is *n*.

Problem 18. Show all legal B-trees of minimum degree 2 that store the keys 1, 2, 3, 4, 5.

Problem 19. Insert the following keys to an empty 5-way B-tree: 3, 7, 9, 23, 45, 1, 5, 14, 25, 24, 13, 11, 8, 19, 4, 31, 35, 56. Draw the result tree after each insertion.

Problem 20. Show the results of deleting C, P, and V, in order, from the following B-tree with $m = 5$:


P
C G M
T X
A B
D E F
J K L
N O
Q R S
U V
Y Z