**Worksheet (tree (BT & BST), heap data structures)**

1. What is the maximum number of nodes in the 8th level of a binary tree?
2. What is the maximum number of nodes that a binary tree of height 8 can accommodate? And how many of them are internal nodes?
3. What is the maximum number of levels that a binary tree with 100 nodes can have?
4. What is the minimum number of levels that a binary tree with 100 nodes can have?
5. A) One hundred integer elements are chosen at random and inserted in to a sorted linked list and a binary search tree. Describe the efficiency of searching for an element in each structure, in terms of Big-O notation.

B) One hundred integer elements are inserted in order from smallest to largest in to a sorted linked list and binary search tree. Describe the efficiency of searching for an element in each structure, in terms of Big-O notation

1. Construct a BST whose elements are inserted in the following order:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 50 | 72 | 96 | 94 | 107 | 31 | 18 | 13 | 15 | 10 | 25 | 51 | 29 | 26 | 95 |

1. What nodes are at level 3?
2. Which levels have the maximum number of nodes that they could contain?
3. What is the height of the tree?
4. What is the maximum height of the binary search tree containing these nodes? Draw such a tree.
5. What is the minimum height of the binary search tree containing these nodes? Draw such a tree.
6. Show the orders in which the nodes in the tree are processed by( refer problem 3)

|  |  |
| --- | --- |
| Breadth first traversal | Depth first traversal |
| 1. Top-down 2. Bottom-up | 1. In Postorder traversal of the tree 2. Inorder traversal of the tree 3. Preorder traversal of the tree |

1. Represent the tree in problem 3 using an array. Assume that the tree to contain only non negative integer.
2. Show how the original tree would look like after the insertion of nodes containing 63, 77, 67 ( refer problem 3)
3. Show how the tree would look like after deletion of 10, 31, 25 and 72.
4. Come up with functions that provide the functionalities, stated below.
5. Given a non-empty **binary search tree**, return the minimum data value found in that tree.
6. Given a non-empty **binary tree**, return the minimum data value found in that tree.
7. Returns the sum of the values of the node in a binary tree.
8. Given a pointer to a node and assuming that the node has a left child, returns a pointer to its inorder predecessor.
9. Repeat d for its inorder successor assuming that the node has a right child.
10. Returns the count of leaf nodes in a binary tree.
11. Returns the count of those nodes that have a single child of a given binary tree.
12. Returns the height of a given binary tree.
13. A function which copies a given binary tree.
14. Display only right children in a binary tree.
15. Repeat e making no assumption
16. Given a non empty binary tree and a sum, return true, if there is a path from the root down to a leaf, such that, adding up all the values along the path equals the given sum.
17. Given two trees, return true if they are structurally identical.
18. Returns true if a given tree is a binary search tree.
19. Implement pre order depth first traversal of a binary tree iteratively. Hint: use stack
20. Implement post order depth first traversal of a binary tree iteratively. Hint: use two stack
21. Implement top-bottom breadth first traversal of a binary tree iteratively. Hint: use queue
22. Implement bottom-top breadth first traversal of a binary tree iteratively. Hint: the sequence generated by **Top-down right to left traversal** is the same as the **reversed** sequence generated by **the bottom-up left to right**
23. Represent the following expression using a binary tree. Comment on the result that you get when this tree is traversed in Preorder, Inorder and postorder. a bc \* d ef \* g h
24. Design an algorithm that constructs a tree for a binary algebraic **postfix** expression (assume double operand).

**Note that it is difficult to construct a tree from its infix binary expression.**

Use stack and the following algorithm:

* When we see an operand, we create a single-node tree and push a pointer to it onto our stack.
* When we see an operator, we pop and merge the top two trees on the stack.

In the new tree, the node is the operator, the right child is the first tree popped from the stack, and the left child is the second tree popped. We then push a pointer to the result back onto the stack.

1. Design an algorithm that evaluates a binary algebraic expression from its binary tree representation.
2. Given the numbers in problem 3.
3. Build a Binary MaxHeap. For each insertion, show the physical (assuming array implementation) and the logical view of the heap.
4. Insert 36, 140 , 78 showing the physical and logical view of the heap in each case
5. Remove the three most urgent items, each time showing the physical and logical view of the heap
6. Given the numbers in problem 3 in an array
7. Assuming the array is the representation of some complete binary tree, draw the tree.
8. Which of the nodes fulfill the heap property and which are not
9. Convert the tree in to a heap using the heapify algorithm
10. Implement siftup and siftdown recursively.
11. Show how to find the kth largest value in a Binary-Heap
12. Give the full implementation of a Binary MinHeap(create(), insert(), DeleteMin(), FindMin()). Little modification is needed on MaxHeap.
13. Suppose you are given k sorted lists, of total length n. Using a heap, show how to merge these into a single sorted list in O(nlogk) time. (Hint: Starting with the case k = 2 can be instructive.)



