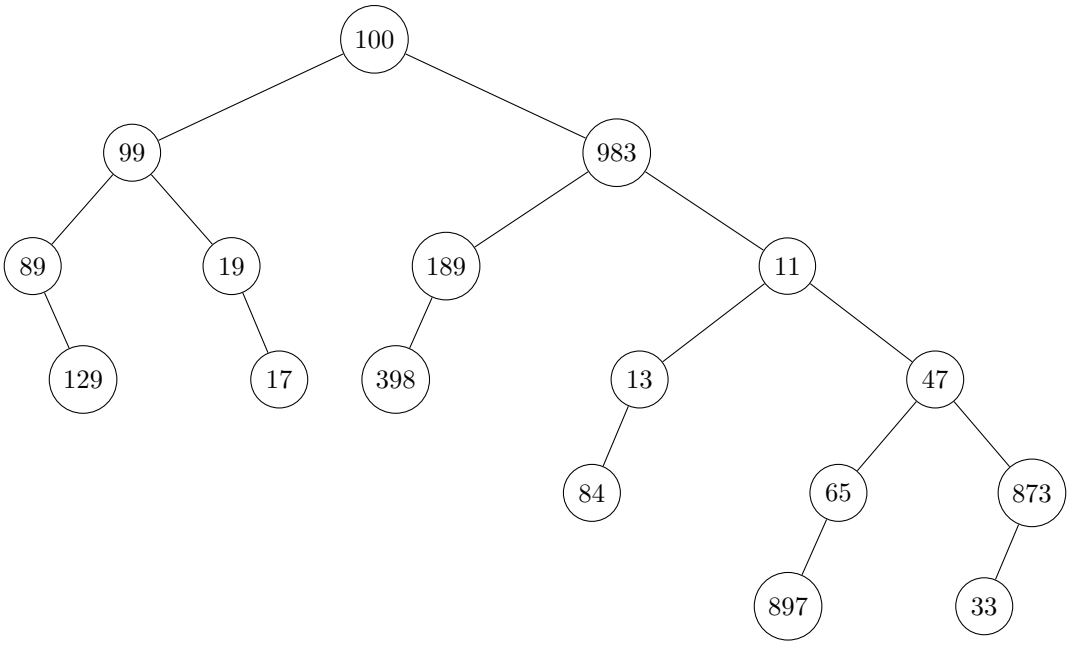

Figure 1: An example binary tree

Binary Trees

1. For the tree shown in Figure 1,
   (a) Which node is the root? _____
   (b) Which nodes are leaves? _____
   (c) What is the tree's height? _____
   (d) What is the result of preorder traversal through the tree?

   .....

   (e) What is the result of postorder traversal through the tree?

   .....

   (f) What is the result of inorder traversal through the tree?

   .....

Binary Heaps and Priority Queues

2. Draw the heap that results when the keys `69 65 83 89 81 85 69 83 84 73 79 78` are inserted, in that order, into an initially empty max-oriented heap. When dealing with equal keys during the swim down operation, take the left branch.

3. Suppose that the sequence `80 82 73 79 * 82 * * 73 * 84 * 89 * * * 81 85 69 * * * 85 * 69` (where a number means insert and an asterisk means remove the maximum) is applied to an initially empty priority queue. Give the sequence of numbers returned by repeated remove the maximum operations.

   .....

   .....

4. A colleague suggests that instead of using a priority queue to implement finding the maximum, you could instead use a linked list, but keep track of the maximmum value inserted so far. Then, when you want the maximum value, you could simply return it, implementing the find the maximum operation in constant time. Why won't this approach work?

   .....

   .....

   .....

   .....

5. For a max-oriented priority queue, suppose that a client calls `insert(...)` with an item that is larger than all items in the queue, and then immediately calls `removeMax(...)`. Is the resulting heap identical to the heap as it was before these operations? Assume that there are no duplicate keys.

   .....

   .....

6. Describe an algorithm to find the smallest 1,000 elements in an unordered array of n integers in $O(n)$ time.

   .....

   .....

   .....

   .....

Binary Search Trees

7. Draw the BST that results when you insert the keys `69 65 83 89 81 85 69 83 84 73 79 78`, in that order, into an initially empty tree. When dealing with equal keys, take the left branch.

8. Suppose that a BST has keys that are integers between 1 and 10, and we search for 5. Which sequence(s) below *cannot* be the sequence of keys examined? There may be more than one correct answer.

   A. 10, 9, 8, 7, 6, 5

   B. 4, 10, 8, 7, 9, 5

   C. 1, 10, 2, 9, 3, 8, 4, 7, 6, 5

   D. 2, 7, 3, 8, 4, 5

   E. 1, 2, 10, 4, 8, 5

9. A colleague decides to sort input data before inserting it into a binary search tree. How will this input sequence affect the runtime of the search operation?

   .....

   .....

   .....

   .....

10. Suppose that we have an estimate ahead of time of how often search keys are to be accessed in a BST, and the freedom to insert them in any order that we desire. Should the keys be inserted into the tree in increasing order, decreasing order of likely frequency of access, or some other order? Explain your answer.

    .....

    .....

11. Using the binary search tree property, devise an algorithm to find the kth smallest and kth largest value in a binary search tree. Expand the algorithm, using the same strategy, to find the largest k and smallest k values in the binary search tree.

    .....

    .....

    .....

    .....

12. Suppose we have a binary search tree where in addition to `left` and `right` pointers, nodes also have `parent` pointers, upwards in the tree to their parents. Using this property, given pointers to two nodes, devise an algorithm to find the nearest common ancestor of the two nodes. For example, given the following tree, node `50` and `63` would have a nearest common ancestor of `52`. However, nodes `23` and `63`

share no ancestors until the root of the tree. In this case, the root, 47, is their nearest common ancestor. Finally, determine the time complexity of your algorithm.