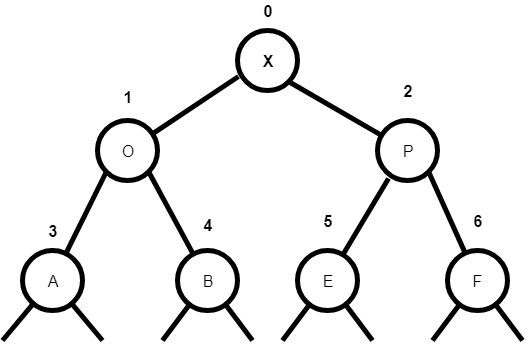
1. (10 Points) Use heap sort to sort the following input array into non-decreasing order.  Ignore array element at index 0.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| Original Array | -1 | 533 | 342 | 714 | 130 | 541 | 10 | 346 | 931 | 57 | 217 |
| After heap construction |  | 931 | 541 | 714 | 342 | 533 | 10 | 346 | 130 | 57 | 217 |
| After Swap & Sink |  | 714 | 541 | 346 | 342 | 533 | 10 | 217 | 130 | 57 | 931 |
| After Swap & Sink |  | 541 | 533 | 346 | 342 | 57 | 10 | 217 | 130 | 714 | 931 |
| After Swap & Sink |  | 533 | 342 | 346 | 130 | 57 | 10 | 217 | 541 | 714 | 931 |
| After Swap & Sink |  | 346 | 342 | 217 | 130 | 57 | 10 | 533 | 541 | 714 | 931 |
| After Swap & Sink |  | 342 | 130 | 217 | 57 | 10 | 346 | 533 | 541 | 714 | 931 |
| After Swap & Sink |  | 217 | 130 | 10 | 57 | 342 | 346 | 533 | 541 | 714 | 931 |
| After Swap & Sink |  | 130 | 57 | 10 | 217 | 342 | 346 | 533 | 541 | 714 | 931 |
| After Swap & Sink |  | 57 | 10 | 130 | 217 | 342 | 346 | 533 | 541 | 714 | 931 |
| After Swap & Sink |  | 10 | 57 | 130 | 217 | 342 | 346 | 533 | 541 | 714 | 931 |
| Final |  | 10 | 57 | 130 | 217 | 342 | 346 | 533 | 541 | 714 | 931 |

2. (10 Points) The input integer array has N entries, which are all non-negative and have equal keys. What is the running time and space complexity for the following sorting algorithms?  Assume each array entry has at most D digits and each digit is within 0 and K.

|  |  |  |
| --- | --- | --- |
|  | Running Time | Space Complexity |
| Merge Sort | O(nlog[n]) | O(n) |
| Quick Sort | O(n2) | O(log[n]) |
| Heap Sort | O(nlog[n]) | O(1) |
| Count sort | O(n + k) | O(k) |
| Radix sort (uses count sort) | O(D[n+k]) | O(n + k) |

3. (6 Points) A binary heap is 0-based, where root is at index 0 and its left child is at index 1 and its right child is at index 2, and so forth. The following figure shows an example with 7 nodes. If a node is at index k, provide the indexes for its parents and children.



|  |  |
| --- | --- |
| Node Index | k (k>0) |
| Parent Index | ceiling(k/2) - 1 (k > 0) |
| Left Child Index | 2k + 1 (k > 0) |
| Right Child Index | 2k + 2 (k > 0) |

Note that if the indexes were numbered as 1, 2, 3… instead of 0, 1, 2… then these algorithms would be different.

4. (4 Points) Draw all of the different binary min-heaps that can be made from the five keys 26, 48, 81, 29, 66.

5. (10 Points) Indicate each of the following statements is true or false.

1. For a binary MinPQ, the maximum key must be at one of the leaves.
2. There are exactly ceiling(n/2) leaves in a binary priority queue with n nodes.
3. An array in decreasing order is a min-oriented heap.
4. In MaxPQ, suppose that a client calls insert() with an item that is larger than

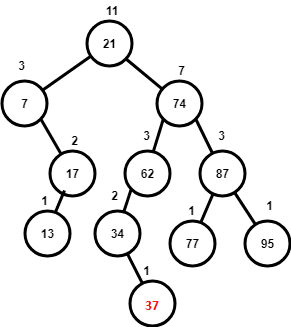
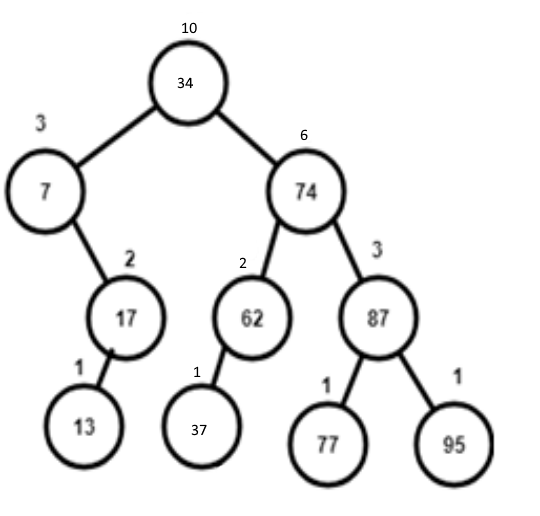
all items in the queue, and then immediately calls delMax(). The resulting heap might not be identical to thnotee heap as it was before these operations.

1. If a node in a BST has two children, its successor has no left child and its predecessor has no right child.

6. (5 Points) 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?

1. 9, 2, 4, 0, 5 NOT IT (0 should never be 4's child, as it's smaller than 2)
2. 6, 1, 2, 4, 5 Valid BST search
3. 4, 6, 9, 10, 5 NOT IT (5 should be to the left of 6. Shouldn't be under 9)
4. 2, 3, 6, 4, 5 Valid BST search
5. 7, 1, 9, 2, 5 NOT IT (9 is bigger than 7 & shouldn’t be 1's child)

7. (5 Points) The following figure shows a BST, where key is displayed inside the node and n value is shown on top of each node. Using the BST delete algorithm provided in class, show the result tree after deleting the root from the following tree. The result tree should show key and n values for each node.

🡪

8. (10 Points) Write a recursive method isBinaryTree() that takes a Node

as argument and returns true if the subtree count field n is consistent in the data structure rooted at that node, false otherwise. Provide your implementation inside **Node** class **isBinaryTree** method.

**Submission Note**

1) For written part of the questions:

1. Write your answers inside a text document (in plain text, MS Word, or PDF format)
2. Name the file as firstname.lastname.assignment3.txt(doc, docx, or pdf) with proper file extension

2) For programming part of the questions

1. Use JDK 1.8 and Junit5
2. Put your full name at the beginning of every source file you created or modified. **2 points will be deducted if your names are not included in the source files.**
3. Do not change the provided package, class, or method name. You can add extra classes or methods if they are needed.
4. **If your code does not compile, you will get zero point**.
5. Use the provided tests to verify your implementation. **Extra tests might be used for grading.**
6. Zip all the source files into firstname.lastname.assignment3.zip

3) Submit both of your files (text document and zip file) via Canvas course web site.

2) Due Oct 14, 11:59 PM