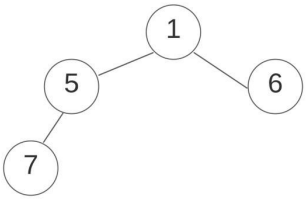
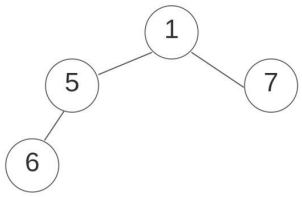
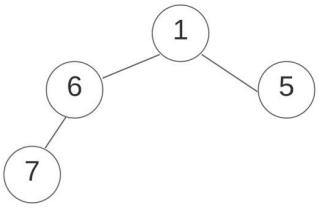
Title: Heaps
Author: Deniz Çalkan
ID: 21703994
Section: 1
Assignment: 3
Description: Answers of questions 1,2  and 4.

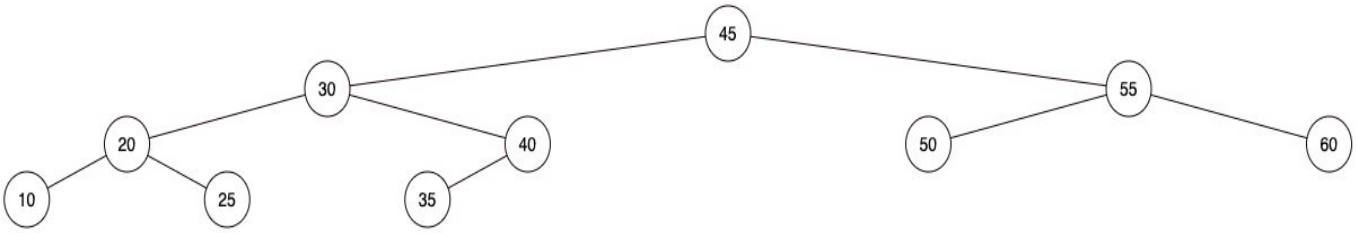
Q1a) All valid min-heaps containing these 4 elements 5, 7, 6, 1.



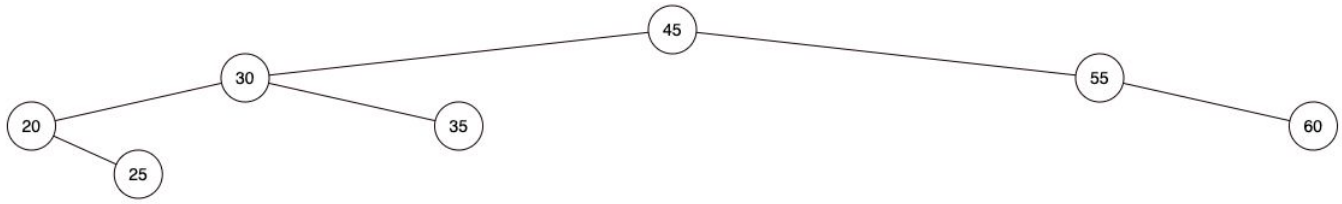




Q1b)  After insertions:



After deletions:




Q2)

| Data Structure | insert | extractMin |
|---|---|---|
| unsorted array | O(1) | O(n) |
| AVL tree | O(logn) | O(logn) |
| min-heap | O(logn) | O(logn) |
| unsorted linked list | O(1) | O(n) |
| sorted linked list | O(n) | O(1) |

Q4)

a) Expected running times of getLessThan and getGreater than methods are O(n) because both methods use preorder traversal and preorder traversal is O(n). Time complexities of the methods can't be enhanced because we need to check all items one by one by comparing with the key. So we cannot find a better running time than O(n).

b) To be able to find the median of a heap in O(1), a min heap and a max heap are used. Since the median method only gets max from max heap and min from min heap its running time is O(1) because the first item in max heap is the biggest item in the heap whereas the first item in the min heap is the smallest item in the min heap. If the sizes of both heaps equal median is (max's max + min's min) / 2, if max heap's size is bigger median is max's max else median is min's min. With each insertion median is changed so each insertion is done by comparing the key with the median then if the key is less than the median the key is inserted to the max heap else min heap. The size difference is kept at most 1 by using the rebalance function. Rebalance function removes the max from max heap and

inserts it to min heap if max heap's size is bigger than min heap size (difference is more than 1)  else the otherwise. By doing this we keep median in the middle. This process is O(logn) since we use insert and delete.