COMP 352 Fall Semester 2017

Assignment 3

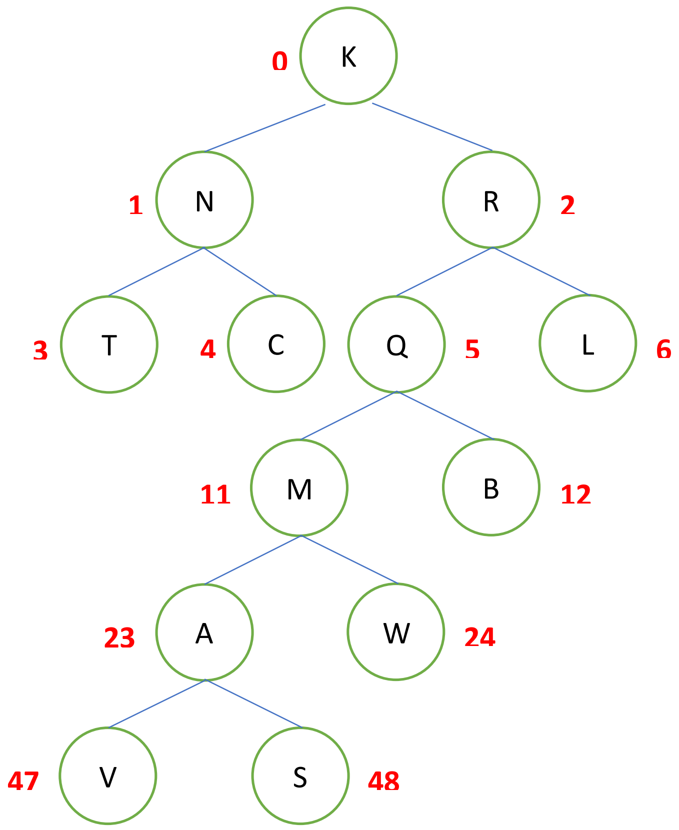
Rui Zhao

40018813

Writing questions:

<u>Question 1:</u>

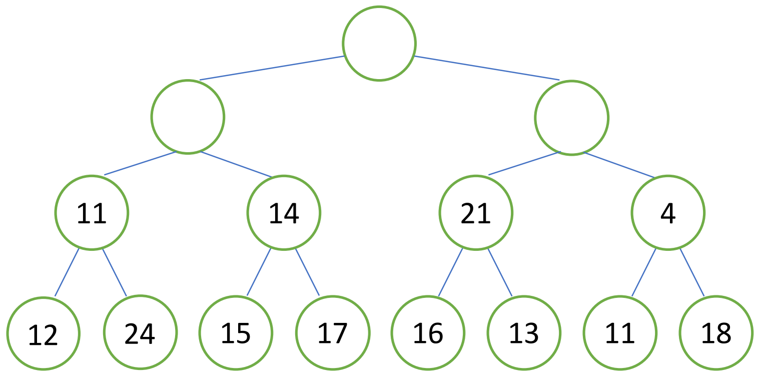
a)

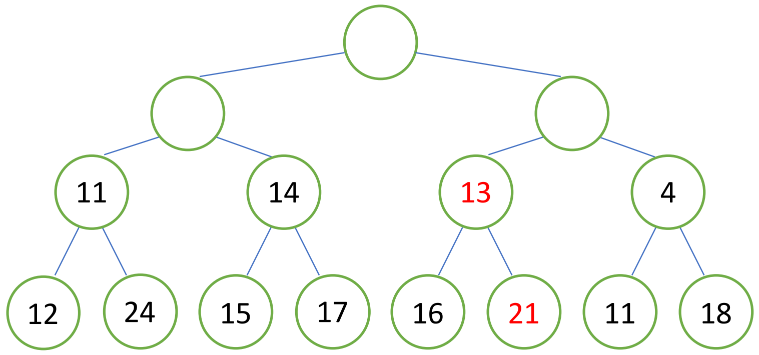


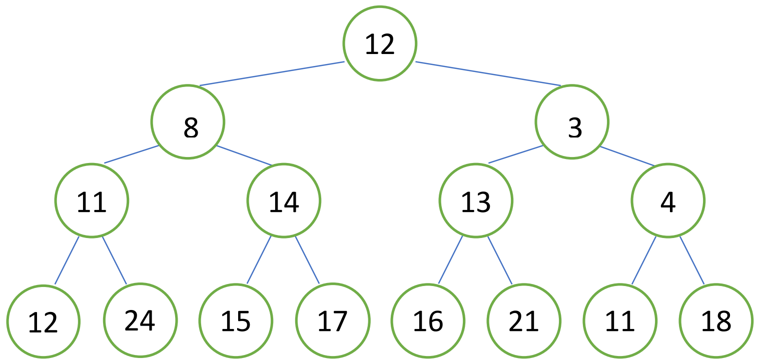
b) For array-based binary tree, the index of left child is 2*index of root +1, and the index of right child is 2*index of root +2. Therefore, the array for storing the given tree is illustrated above. The red numbers indicate the indices of the array in which corresponding values are stored.

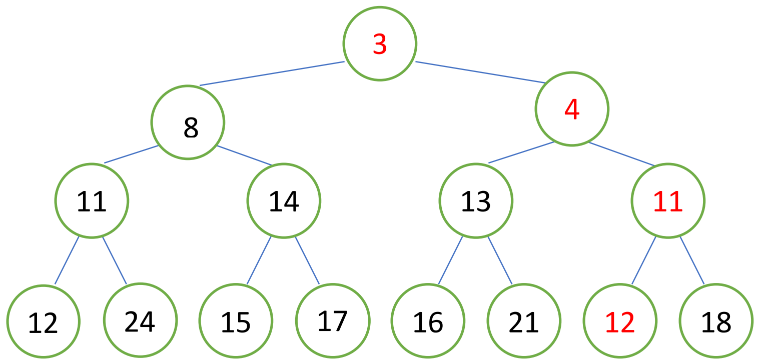
Question 2:

a)

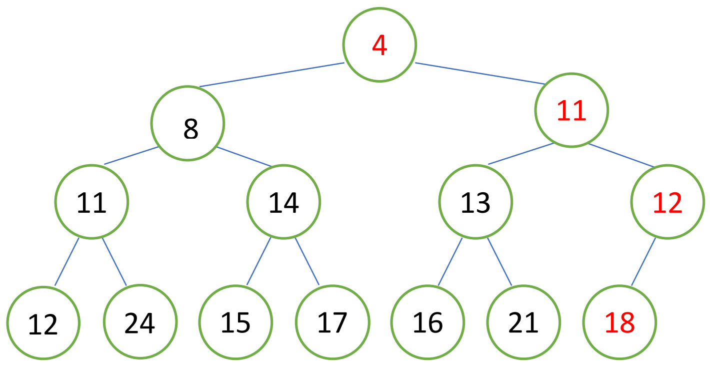




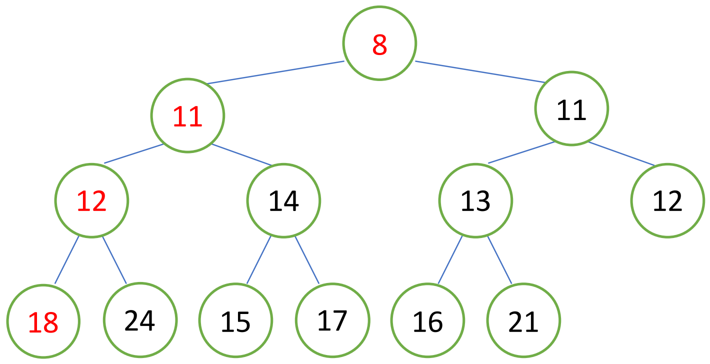




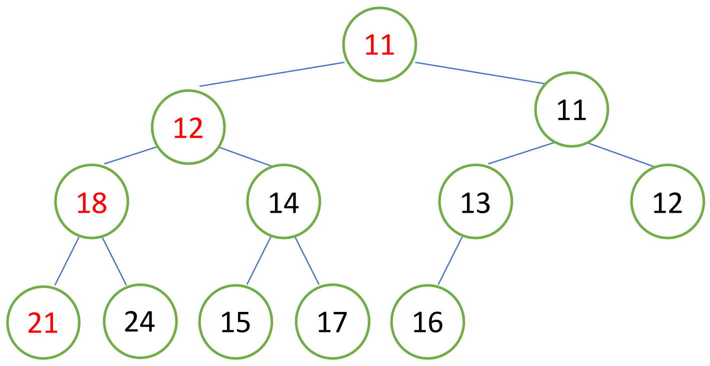
After 1st removeMin



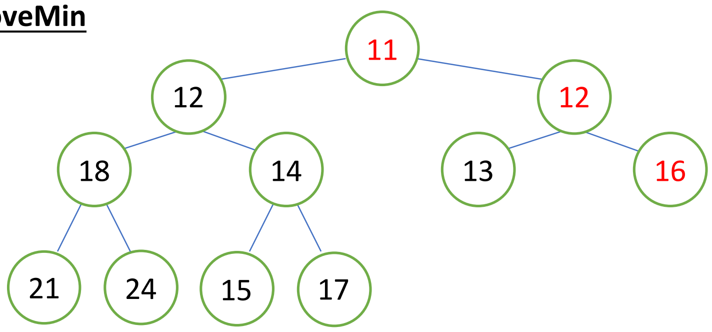
After 2nd removeMin



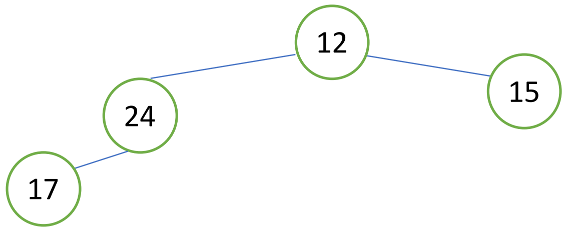
After 3rd removeMin

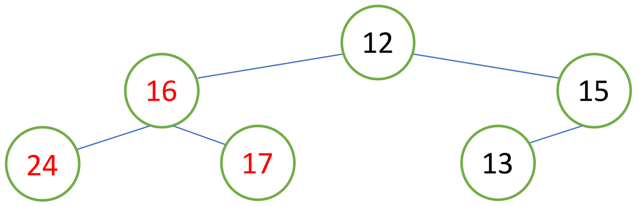


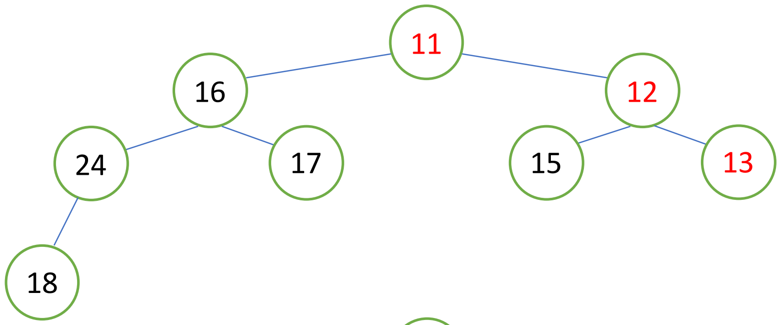
After 4th removeMin

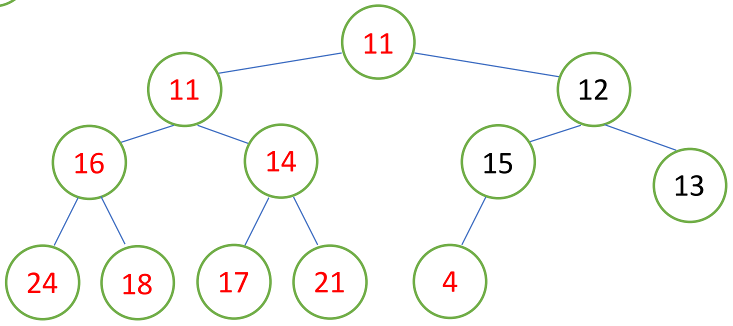


b)

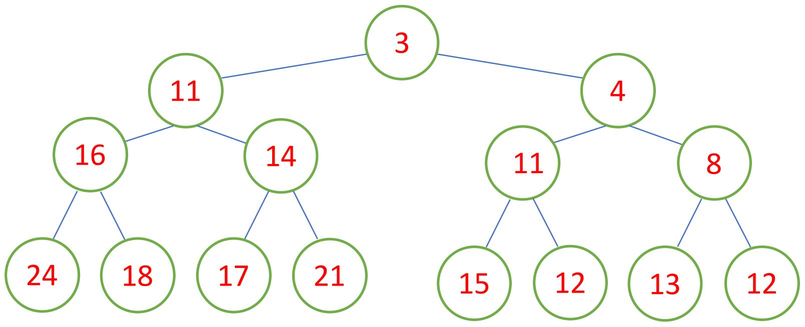








<u>Final tree (some steps are omitted):</u>



<u>Question 3:</u>

Algorithm computeDepth(Tree T, Node v)

Input: tree T with n nodes, v is a node of T

Output: the depth of each node in the subtree of T rooted at v

```
if T.isRoot(v) then
   setDepth(v,0)
else then
   setDepth(v,1+getDepth(T.parent(v)))
children ← T.children(v)
while (childen.hasNext()) do
    computeDepth(T, children.next())
```

a) the time complexity of this algorithm is O(n), because it is a linear recursion function and will make n recursive calls in total.

b) The best possible complexity that can be achieved to compute the depth of all nodes of the tree T should be O(n), because we have to visit each node at least once in order to computer its depth.

c) My algorithm is O(n) in terms of time complexity.