LM Data Structures, Algorithms, and Databases (34140, 34141, 34139, 36989)

Exercise Sheet Solution

Week 03

Q1. MCQs

1.1.

$f(n)$ is $O(g(n))$ means, intuitively, $f(n)$ is approximately proportional to $g(n)$ for large values of *n*. Which of the following statements is true?

A $5n^2 + 8n - 20$ is $O(n^2)$

B $n + \log\ n$ is $O(n)$

C $n \cdot \log\ n$ is $O(n)$

D $4n^2 - 256\,n$ is $O(n)$

Answer: A, B

1.2

$f(n)$ is $O(g(n))$ means that $f(n)$is bounded ***above*** by some constant times $g(n)$ for large values of *n.* Which of the following statements is true?

A $8n - 20$ is $O(n^2)$

B $n + \log\ n$ is $O(n)$

C $n \cdot \log\ n$ is $O(n^2)$

D $5n^2 + 8n$ is $O(n)$

Answer: A, B, C

1.3 What is the worst-case time complexity for insertion in a binary search tree with n elements?

A $O(\log\ n)$

B $O(n)$

C $O(n^2)$

D $O(n\ \log\ n)$

Answer: B

1.4. What is the worst-case time complexity for insertion in an AVL tree with n elements?

A $O\left(\log\ n\right)$

B $O\left(n\right)$

C $O\left(n^2\right)$

D $O\left(n\ \log\ n\right)$

Answer: A

Q2. Determine the time complexity for the given algorithms

2.1 An algorithm which multiplies all elements in the array:

```
def product(arr):
    n = len(arr)
    x = 1
    i = 0
    while i < n:
        x *= arr[i]
        i += 1
    return x
```

Answer: O (n)

2.2 An algorithm which modifies the last value in the array

```
def modify(arr):
    if len(arr) == 0:
        raise Exception("Array is empty")

    last = arr[-1]

    if last < 0:
        last = -last

    arr[-1] = last
```

Answer: O (1)

2.3 Finding the largest element of the array (method 1)

```
def largest1(arr):
    n = len(arr)
    max_val = 0

    for i in range(n):
        is_largest = True

        for j in range(n):
            if arr[i] < arr[j]:
                is_largest = False

        if is_largest:
            max_val = arr[i]

    return max_val
```

Answer: O (n^2)

2.4 Finding the largest element of the array (method 2)

```
def largest2(arr):
    n = len(arr)
    max_val = 0

    if n == 0:
        return 0
    else:
        max_val = arr[0]
        for i in range(n):
            if arr[i] > max_val:
                max_val = arr[i]

        return max_val
```

Answer: O (n)

2.5 Finding the largest element of the array (method 3)

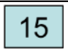
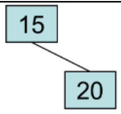
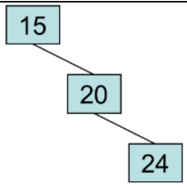
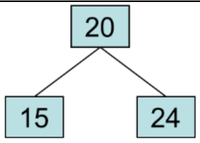
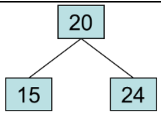
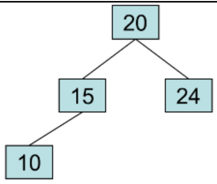
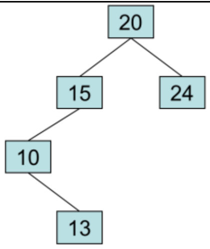
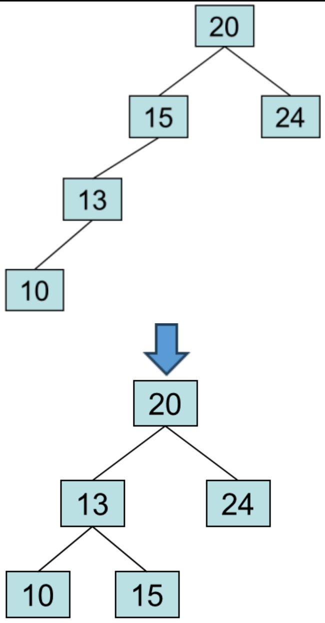
```
def largest3(arr):
    arr.sort() # assume O(n log n)

    if len(arr) == 0:
        return 0
    else:
        last = arr[-1]
        return last
```

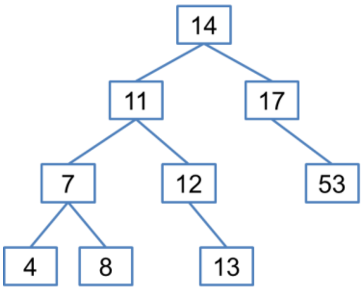
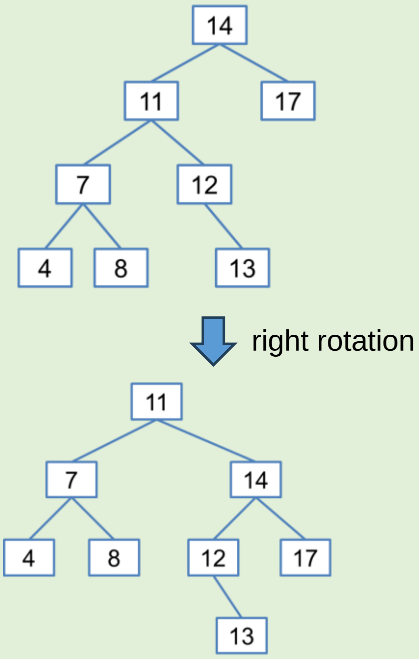
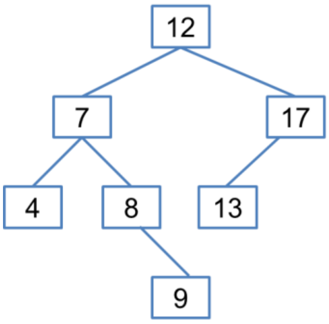
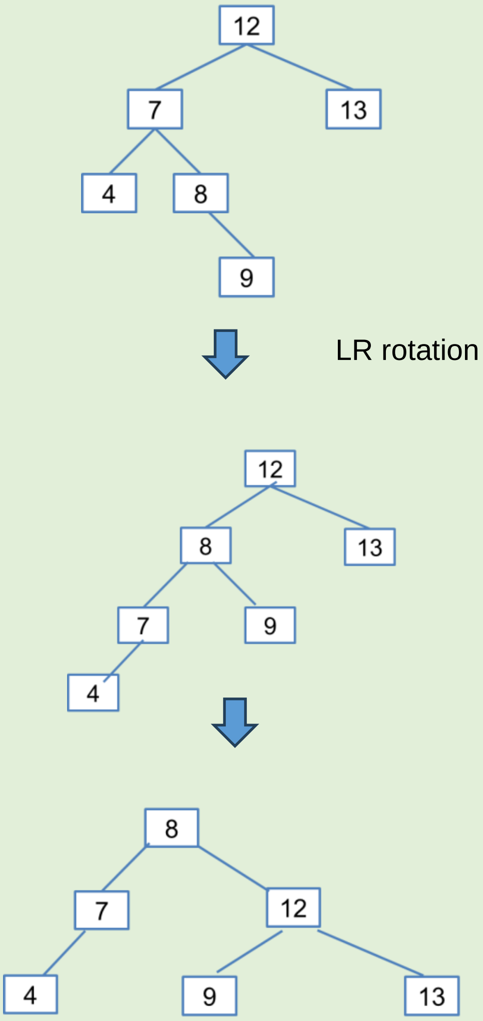
Answer: O (n log n)

Q3. Build an AVL tree with the following values: 15, 20, 24, 10, 13.

Answer:

15			
15, 20			
15, 20, 24		Left Rotation	
15, 20, 24			
15, 20, 24, 10			
15, 20, 24, 10, 13		LR Rotation	

Q4. Show the result of deleting values from the given AVL tree.

 	Delete 53	 14 11 17 7 12 4 8 13 right rotation 11 7 14 4 8 12 17 13
 	Delete 17	 12 7 13 4 8 9 LR rotation 12 8 13 7 9 4 8 7 12 4 9 13

Q5. Create pseudocode for a function called **calculateSum** that takes the root of a binary tree as input and computes the sum of all the numbers stored in the nodes of the tree.
Answer:

```
function calculateSum(Node root)
    if root is null
        return 0

    leftSum = calculateSum(root.left)
    rightSum = calculateSum(root.right)

    return root.val + leftSum + rightSum
```

Q6. Create pseudocode for a function called **nodeAtLevel(tree, theLevel)**. This function should return null if the binary tree does not contain any nodes at level theLevel; otherwise, it should return the nodes present at this level.

```
    1
   / \
  2   3
 / \   \
4   5   6
         \
          7
```

For instance, given the following tree, when called with **nodeAtLevel(root, 3)**, the function should return [4, 5, 6].

Your task is to design the pseudocode for the **nodeAtLevel** function. What is the time complexity of your code as a function of the number of nodes in the binary tree?

Answer: The time complexity of the provided code is O(n) in the worst case, as it may visit every node of the binary tree once.

```
function nodeAtLevel(tree, theLevel):
    if tree is null:
        # empty tree, no node at level theLevel
        return null

    if theLevel equals 1:
        # tree is at level 1
        return tree.Value

    # search for desired node in left subtree
    x = nodeAtLevel(tree.leftChild, theLevel - 1)

    if x is not null:
        # found an node at level theLevel
        return x

    # return desired node from right subtree
    return nodeAtLevel(tree.rightChild, theLevel - 1)
```

Q7. Illustrate the binary tree structure with single-character data fields, given the inorder traversal output as ABCDEFGHIJ and postorder traversal output as BDCAEHGJIF.

Answer:

```
           F
       /      \
     E         I
  /           /  \
A           G     J
 \           \
   C          H
 /  \
B    D
```