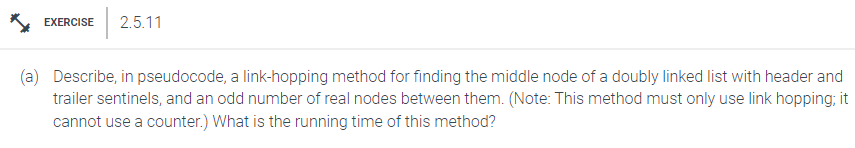
# Michael Chillemi

# 06/02/2023

# CS 590 - Algorithms

# M2.B2: Module 2 Basic Data Structures Creativity Exercises

Problem 2.5.11



Answer:

Algorithm printMiddleNode(doublyLinkedList):

Input:doublyLinkedList

Output: None, Method will provide a print out to the console.

slowPointer = doublyLinkedList.head

fastPointer = doublyLinkedList.head

While fast and fast.next

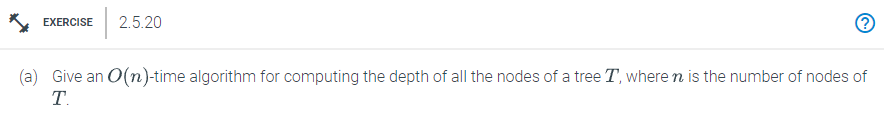
slowPointer = slowPointer.next

fastPointer = fastPointer.next.next

Print(“Middle Node = ”, slowPointer.data)

The time complexity of this algorithm is . This is because n is equal to the number of nodes in the doubly linked list and in this algorithm we just loop over the doubly linked list once.

Problem 2.5.20



Answer:

Algorithm maxDepth(node, current\_depth):

Input: node, current\_depth

Output: max\_depth

If node is null

Return max\_depth

current\_depth++

If current\_depth > max\_depth

max\_depth = current\_depth

maxDepth(leftnode,current\_depth)

maxDepth(rightnode,current\_depth)

An approach for doing this is a pre-order traversal. Whenever we visit a node in the traversal, we simply record the maximum depth, which is the current depth plus 1. Once the algorithm visits every node then it will return the max depth. The running time for this algorithm is . This is the case because we visit every node which is the length of the tree.