

# 112Exam2

Wednesday, November 13, 2019      6:36 PM



CS112OldMi  
dtermExa...

CS 112

MidtermExam A

65 minutes

Closed Book and no Aid allowed

15th November 2017

Instructor: Abbas Attarwala

Boston University

**Note:** It is a violation of the Honor Code to discuss this midterm exam question with anyone until after everyone in CS 112 (A and B section) has taken the exam.

Enter your First Name: \_\_\_\_\_

Enter your Last Name: \_\_\_\_\_

Enter your Student#: \_\_\_\_\_

"I pledge my honor that I have not violated the Honor Code during this examination."

Your Signature: \_\_\_\_\_

Question:	1	2	3	4	Total
Points:	13	8	20	0	41
Bonus Points:	0	0	0	3	3
Score:					

DO NOT TURN THIS PAGE UNTIL YOU HAVE BEEN ASKED TO DO SO



- Given the following Linked List class<sup>1</sup>. You will find the sub-questions on the next page.

```
public class LinkedList<E extends Comparable<E>> {
    private static class Node<E>
    {
        private E data;
        private Node<E> next;

        public Node(E d)
        {
            data=d;
            next=null;
        }
    }
    private Node<E> head;
    private Node<E> tail;

    public boolean increasingSequence() throws EmptyLinkedList
    {
        //TODO: Complete the rest of this method.
    }
}
```



```

    }

    /*
     * n must be non-null when _increasingSequence is called.
     * You must use recursion here.
     */
    private boolean _increasingSequence(Node<E> n)
    {
        if (n.next==null)
            return true;
        Node<E> previous=n;
        Node<E> next=n.next;
        //TODO: Complete the rest of this method.

    }

}

```

---

<sup>1</sup>The template for the `LinkedList` class is exactly the same as seen in lecture

- (a) Complete the method `increasingSequence2` such that the method returns back a `true` when the `LinkedList` is in an increasing order and returns back `false` when the `LinkedList` is not in an increasing order. This method returns back an exception `EmptyLinkedList3` when the `LinkedList` is empty (i.e. head points at `null`). You can call the `compareTo` method to compare two `Nodes`. We provide the following 9 `LinkedLists` and their expected output.

 <code>3 --&gt; 7 --&gt; 5 --&gt; null</code>	<i>must return FALSE</i>
<code>3 --&gt; 4 --&gt; 5 --&gt; 6 --&gt; null</code>	<i>must return TRUE</i>
<code>3 --&gt; 5 --&gt; 7 --&gt; 9 --&gt; null</code>	<i>must return TRUE</i>
<code>3 --&gt; 3 --&gt; 3 --&gt; null</code>	<i>must return FALSE</i>

\_\_\_\_\_

\_\_\_\_\_

(7)

<code>3 -- &gt; 4 -- &gt; 5 -- &gt; 7 -- &gt; null</code>	<i>must return TRUE</i>
<code>6 -- &gt; 5 -- &gt; 4 -- &gt; 3 -- &gt; null</code>	<i>must return FALSE</i>
<code>- 1 -- &gt; 0 -- &gt; null</code>	<i>must return TRUE</i>
<code>3 -- &gt; null</code>	<i>must return TRUE</i>
<code>&lt; EMPTYLINKEDLIST &gt;</code>	<i>must throw EmptyLinkedList Exception</i>

You can use the space provided in the code to write your answer.

## On .java file

- (b) Write (do not solve) the recurrence  $T(n)$  for the `_increasingSequence`.

$$\Downarrow T(n) = \underbrace{T(n-1)}_1 + \underbrace{C}_1$$

- (c) Solve the recurrence  $T(n)$  for the `_increasingSequence` from [b] above, in the worst case and find the  $\mathcal{O}(\dots)$ . Show us all the steps and insert your final  $\mathcal{O}(\dots)$  answer by drawing a box around it.

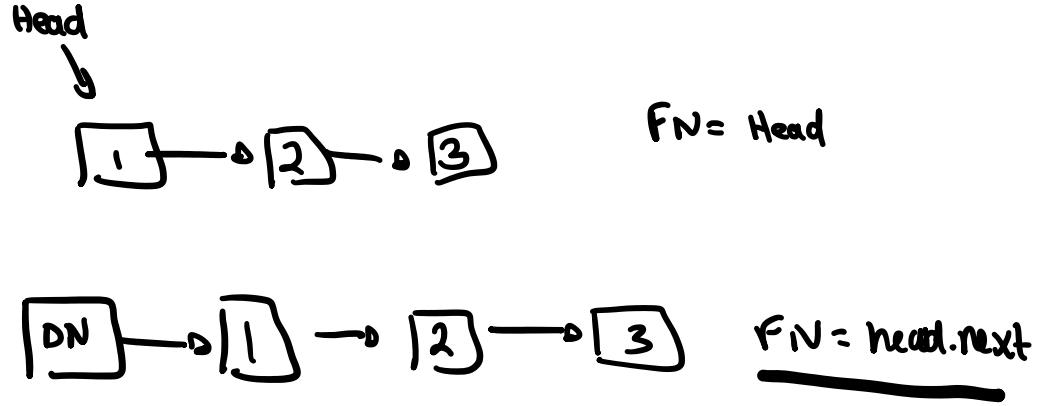
$$\begin{aligned}
 T(n) &= T(n-1) + C \quad \textcircled{1} \\
 &= [T(n-2) + C] + C = T(n-2) + 2C \quad \textcircled{2} \\
 &= [(T(n-3) + C) + C] + C = T(n-3) + 3C \quad \textcircled{3} \\
 &\vdots \\
 &= T(n-i) + i \cdot C \\
 &= T(n-n) + n \cdot C = T(0) + n \cdot C = 0 + n \cdot C = C
 \end{aligned}$$

Total for Question

<sup>2</sup>This problem is very similar to `removeDuplicates` from review session.

<sup>3</sup>You do not have to create this class

2. Given the following general tree. There is no coding required for this question.



$\Theta(n)$

1: 13

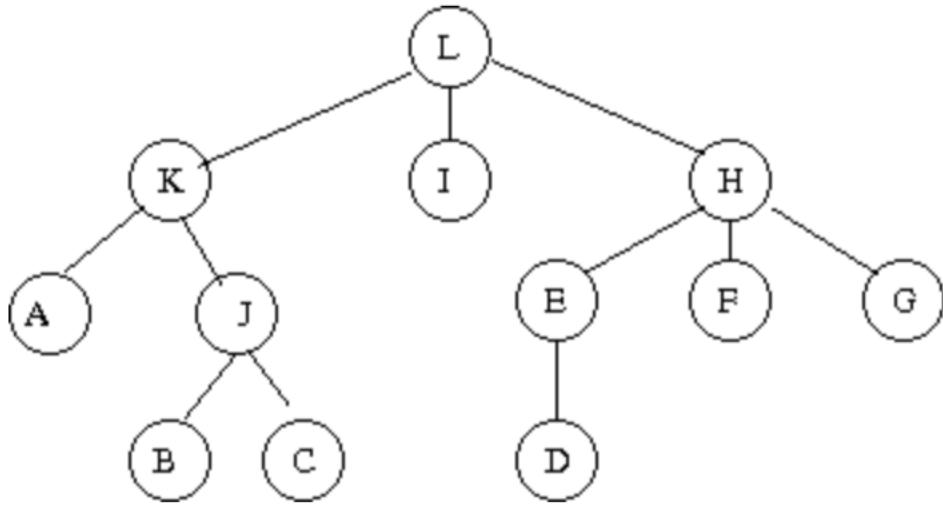


Figure 1: Tree with 12 nodes

(a) What is PreOrder traversal on the above tree?

LKAJBCIHEDFG

(b) What is PostOrder traversal on the above tree?

ABCJKIDFGCHL

(c) What is breadth-first order traversal or levelwise order traversal on the above tree?

L KIH AJEFG BCD

(d) How many leaf nodes does the above tree contain?

7

(e) What is the height of the above tree?

4

Total for Question

(2)

(2)

(2)

(1)

(1)

on 2: 8

3. Given the following `BinaryTree` class

```
public class BinaryTree<T> {
    private static class Node<T>
    {
        private int data;
        private Node<T> left;
        private Node<T> right;

        public Node(int d)
        {
            data=d;
        }
    }
    private Node<T> root; //points at the root of the tree

    /* Returns the height of the tree. The height of an empty tree OR
     * a tree with just a single node is 0. You must use recursion
     * on _getHeight to answer this
     * question. */
    public int getHeight()
    {
        //TODO
    }

    private int _getHeight(Node<T> r)
    {
        //TODO
    }
}
```



```
}
```

```
/* Returns the clone of BinaryTree represented by the reference 'this'  
 *Pay careful attention to the return types of clone  
 *and _clone. */  
public BinaryTree<T> clone ()  
{  
    BinaryTree<T> ret=new BinaryTree<>();  
    //TODO  
  
}  
private Node<T> _clone(Node<T> r)  
{  
    if (r==null)  
        return null;  
    //TODO  
  
}  
}
```

- (a) In the lecture, we calculated the height of a general Tree using recursion. You are now asked to find the height of a BinaryTree<sup>4</sup>. You will use recursion and complete getHeight and \_getHeight. You can use the space in the code to answer this question.

(7)

## On .java file

- (b) In your assignment3 you cloned a Set and then in your assignment6 you cloned a LinkedList.  
In this question, we ask you to clone a BinaryTree. You will use recursion and complete `clone` and `_clone`. You can use the space in the code to answer this question.

on .java file.

<sup>4</sup>Hint: You can use the `max` function from the `Math` class

Page 6 of 10

- (c) In the main function provided, write code so that you create the following `BinaryTree`<sup>5</sup> using the above `BinaryTree` class.

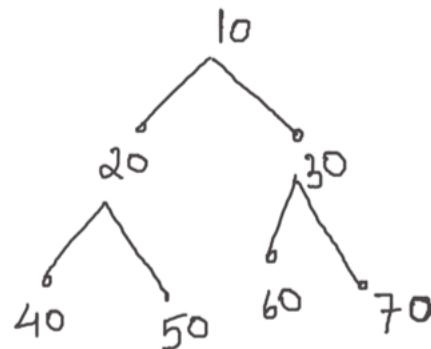


Figure 2: BinaryTree with 7 nodes

```
//You can assume that this main function belongs inside the above  
//BinaryTree class.  
public static void main(String[] args)  
{
```

(7)

(6)

}

Total for Question

---

<sup>5</sup>In lecture we created similar trees by hardcoding values to test various algorithms. You can use similar technique when answering this question

Page 7 of 10

4. (a) Given the following code<sup>6</sup>:

```
for (i=1; i<=n; i++)  
    for (j=1; j<=i; j++)  
        k=k+1;
```

(1 1/2 (b))

You are asked to count the actual number of times the instruction  $k = k + 1$  is executed and then find the  $\mathcal{O}(\dots)$ . Your friend from NorthEastern performs the following series of steps:

$$\begin{aligned}&= \sum_{i=1}^n \sum_{j=1}^i 1 \\&= \sum_{i=1}^n n \\&= n \sum_{i=1}^n 1\end{aligned}$$

n 3: 20

onus))

$$= n * n$$

$$= n^2$$

i.e.  $n^2$  is  $\mathcal{O}(n^2)$ . The final  $\mathcal{O}(n^2)$  answer is correct. However, there is a flaw in the above steps. What is the flaw and repeat the steps<sup>7</sup> such that the flaw is corrected.

$$\begin{aligned}
 &= \sum_{i=1}^n \sum_{j=1}^i 1 \\
 &= (1) \cdot 1 + (2) \cdot 1 + \dots + (n) \cdot 1 \\
 &= 1 + 2 + 3 + \dots + n \\
 &= \frac{n(n+1)}{2} = O(n^2)
 \end{aligned}$$

<sup>6</sup>Similar to Piazza post @199

<sup>7</sup>You must also use the  $\sum$  in your step(s) similar to what your friend did.

- (b) What is the  $\mathcal{O}(\dots)$  for the following piece of code? Make sure that you explain your final answer!  $1 \frac{1}{2}$  (b)

```
public void someFunction(int n){  
    int i,j,k,count=0;  
    for (i=n/2;i<=n;i++) ← O(n²) = O(½·n) = O(n)}
```

onus))

```

{
    for (j=1; j<=n; j=2*j) ←  $O(\log_2 n)$ 
    {
        for (k=1; k<=n; k=k*2) ←  $O(\log_2 n)$ 
        {
            count++;
        }
    }
}

```

$$1 \rightarrow 2 \rightarrow 4 \rightarrow 8$$

$$\begin{aligned}
&= O(n) \cdot O(\log n) \cdot O(\log n) \\
&= O(n \cdot \log n \cdot \log n) \\
&\leq O(n(\log n)^2)
\end{aligned}$$

$$\begin{array}{l}
\text{for } i = 1 \text{ to } n \\
\quad \text{for } j = 1 \text{ to } n
\end{array} \quad O(n) + O(n) = O(2n) = O(n)$$

$$O(n) \cdot O(n) = O(n^2)$$



**Extra Rough Page**





