NJIT New Jersey Institute of Technology

Number on the list	Grade

Midterm Exam

CS435-003

Time: 70 minutes

Do not remove the top-left staple. Your papers must remain attached during the exam.

Please write your name once you get seated.

Name:

Instruction:

1. Smart devices (e.g., cellphones, smartwatches, AirPods) are not allowed during the exam. Please ensure they are turned off and place your cellphone in the designated area before the exam begins.
2. Students may not leave the exam room for any reason while the exam is in progress.
3. The exam is closed book. The use of study materials, laptops, notes, or textbooks is strictly prohibited.
4. Backpacks and personal belongings (e.g., jackets, laptops) must be placed at the front of the classroom before the exam begins. Access to your belongings during the exam is not permitted, so please ensure you have everything you need on your desk before the exam starts.
5. Please verify that the number on this exam paper matches your assigned number and that you are seated in the correct location.
6. You may not use your own paper during the exam. If you need extra paper, please request it from the instructor or proctors.
7. Please write clearly. Illegible handwriting may result in lost marks.
8. If you erase or cross out any part of your solution and then rewrite it, the rewritten portion will not be eligible for **re**grading. Please ensure your final answers are clear and unaltered to facilitate accurate grading.

1. a) Use the Master Theorem to find the asymptotic solution of the following recurrence relation. (5 points)
b) Solve the following recurrence relation using the expansion method. (20 points).

$$T(n) = T\left(\frac{n}{2}\right) + n, \qquad T(1) = 2$$

a) $a = 1, b = 2, k = 1: a = 1 < 2^1 = b^k$
$p = 0, then\ T(n) = O(n^1 \log^0 n) = O(n)$

b)

$$T(n) = T\left(\frac{n}{2}\right) + n = \left(T\left(\frac{n}{2^2}\right) + \frac{n}{2}\right) + n$$

$$T(n) = T\left(\frac{n}{2^2}\right) + \left(\frac{n}{2} + n\right) = \left(T\left(\frac{n}{2^3}\right) + \frac{n}{2^2}\right) + \left(\frac{n}{2} + n\right) = T\left(\frac{n}{2^3}\right) + \left(\frac{n}{2^2} + \frac{n}{2} + n\right)$$

$$= \left(T\left(\frac{n}{2^4}\right) + \frac{n}{2^3}\right) + \left(\frac{n}{2^2} + \frac{n}{2} + n\right) = T\left(\frac{n}{2^4}\right) + \left(\frac{n}{2^3} + \frac{n}{2^2} + \frac{n}{2} + n\right)$$

$$T(n) = T\left(\frac{n}{2^k}\right) + \left(\frac{n}{2^{k-1}} + \cdots + \frac{n}{2^2} + \frac{n}{2} + n\right) = T\left(\frac{n}{2^k}\right) + n\left(\frac{1}{2^{k-1}} + \cdots + \frac{1}{2^2} + \frac{1}{2} + 1\right) =$$

$$= T\left(\frac{n}{2^k}\right) + n\left(2 - \frac{2}{2^k}\right)$$

Setting $2^k = n$:

$$T(n) = T\left(\frac{n}{2^k}\right) + n\left(2 - \frac{2}{n}\right) = T(1) + 2n - 2 = 2 + 2n - 2 = O(n)$$

2. a) A research lab is testing two different data transmission methods, **Method A** and **Method B**, for sending large amounts of scientific data from space telescopes to Earth. The number of data packets to be transmitted is n, which can reach up to 10^{12}. The average transmission times (in microseconds) are given by: $\boldsymbol{T_A(n) = 15n \log_{10} n}$ and $\boldsymbol{T_B(n) = nlog_{10}^2 n}$. In the **Big-O sense**, which method is asymptotically more efficient? (5 points) Show that Method B is faster in this problem, and justify your answer. (5 points)

b) Determine the time complexity of the following code. (10 points).

```
FOR i = 1 TO n STEP i = i * 2 DO
    FOR j = 1 TO i DO
        Perform an O(1) operation
```

a) $\boldsymbol{T_A(n) = O(n\, log\, n)}$ and $\boldsymbol{T_B(n) = O(nlog^2 n)}$

Algorithm **A** is **asymptotically** better because $\boldsymbol{nlogn < nlog^2\, n}$

$T_B(n) < T_A(n) \Rightarrow nlog_{10}^2 n < 15n \log_{10} n \Rightarrow \log_{10} n < 15 \Rightarrow n < 10^{15}$

The lab states n can reach up to 10^{12}. Since $10^{12} < 10^{15}$, every n in the lab's range satisfies $T_B(n) < T_A(n)$, so Method B is faster in this problem's range.

b) O(n)

$$1 + 2 + 2^2 + \cdots + 2^{logn} = 2^{\log n+1} - 1 = 2n - 1$$

3. Answer the following questions. (30 points)

a) Consider an array A of length n and a function `func(.)` with quadratic time complexity. Find the time complexity of `mystery(A,n)`: (Use the Master Theorem if applicable.)

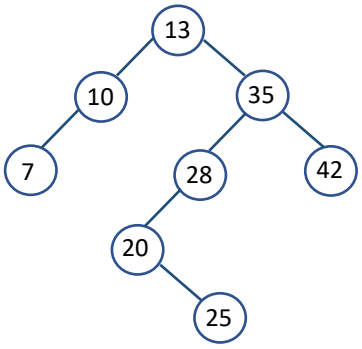
```
mystery(A,n):
    IF n==1 THEN
        RETURN 1
    func(A)
    RETURN mystery(A,n-1)+n
```

$T(n) = T(n-1) + O(n^2): T(n) = O(n^3)$

b) Write a recursive function for preorder traversal of a binary tree and perform a postorder traversal of the following tree and write the sequence of nodes visited.

```
postorderTraversal(node):
    If node == null Then
        return
    postorderTraversal(node.left)
    postorderTraversal(node.right)
    print(node.key)  # Visit the root
```

7,10, 25, 20,28,42,35,13



c) What is the time complexity of searching for a value in a general binary tree? Is this search complexity optimal? If not, under what conditions does a general binary tree achieve optimal search efficiency

O(n), No, For a general binary tree, there is no search algorithm faster than O(n) in the worst case.

d) Suppose you have a **Trie** that stores **n valid English words**, including `"book"`, `"books"`, and `"bookshelf"`. How many operations are needed to **delete the word "books"** from this Trie? Explain your answer. 5 operations, because we only traverse the nodes corresponding to "books" and unmark the end-of-word flag; we don't delete the s node since it's used by "bookshelf."

e) What is the maximum height of a Binary Search Tree (BST)? What kind of input can cause the BST to reach this maximum height?

Maximum height = n−1, Caused by inserting **already sorted data** into the BST.

4. You are given the root of a **Binary Tree**.

(a) Write a **recursive function** to find the **number of nodes in the subtree rooted at a given node**. Also, state the **time complexity** of your solution. (10 points)

(b) Modify your solution from part (a) to compute the **number of nodes in the subtree rooted at every node** in the tree **efficiently**. (15 points)

a)

```
def countSubtreeNodes(node):
    if node is None:
        return 0
    left_count = countSubtreeNodes(node.left)
    right_count = countSubtreeNodes(node.right)
    return 1 + left_count + right_count
```

Time complexity is O(n)

b) by storing the calculated values we can maintain the time complexity

```
def computeAll(node, size_map):
    if node is None:
        return 0

    left_size = computeAll(node.left, size_map)
    right_size = computeAll(node.right, size_map)

    size = 1 + left_size + right_size
    size_map[node] = size

    return size
```

we need to call it for root to calculate all

```
A={}
computeAll(root, A)
```

Time complexity is O(n)

Hints:

- The Master Theorem:

1) $T(n) = a\,T(n-b) + f(n),\ \ f(n) = \mathrm{O}(n^k),\ \ a > 0,\ b > 0,\ k \geq 0$

1. $if\ a = 1, T(n) = O\,(nf(n))$
2. $if\ a > 1, T(n) = O\left(a^{\frac{n}{b}} f(n)\right)$
3. $if\ a < 1, T(n) = O\,(f(n))$

2) $T(n) = a\,T\left(\frac{n}{b}\right) + f(n),\ \ f(n) = \theta(n^k \log^p n), a > 0, k \geq 0, b > 1$

1. $if\ a > b^k, then\ T(n) = \theta\left(n^{\log_b a}\right)$
2. $if\ a = b^k, then$

$(a)\ if\ p > -1, then\ T(n) = \theta\left(n^{\log_b a} \log^{p+1} n\right)$
$(b)\ if\ p = -1, then\ T(n) = \theta\left(n^{\log_b a} \log\log n\right)$
$(c)\ if\ p < -1, then\ T(n) = \theta\left(n^{\log_b a}\right)$

3. $if\ a < b^k, then$

$(a)\ if\ p >= 0, then\ T(n) = \theta(n^k \log^{\mathrm{p}}\ n)$
$(b)\ if\ p < 0, then\ T(n) = \theta(nk)$

- Q1:

$$\sum_{i=0}^{m} \frac{1}{2^i} = 2 - \frac{2}{2^{m+1}}$$

- Q2:

$$\sum_{i=0}^{m} 2^i = 2^{m+1} - 1$$