National University of Computer and Emerging Sciences 0712

Lahore Campus

D & A of Algorithms (CS2009)

Sessional-I Exam

Total Time (Hrs): 1
Total Marks: 15
Total Questions: 3

Date: Sep 22 2025

Course Instructor(s)

Dr. MB, Dr. SK, Dr. MB, Dr. MAQ, AA, UH, SK

Roll No Section Student Signature

6+2+2

Instructions: Answer in the space provided. Do not attach rough sheets with this exam.

CLO 2: **Analyze** the time and space complexity of different algorithms by using standard asymptotic notations for recursive and non-recursive algorithms.

Question 1: [2+ 2 + 3 = 7 Marks]

$\approx 2^{n+5} + 2^{2n+4} + 2^{4N+2}$

(a) $f(n) = 30*2^n + 15*4^n + 3*16^n$

Which of the following statements are true about f(N)

i. $f(n) = O(4^{2n})$
ii. $f(n) = O(8^{n+2})$
iii. $f(n) = O(4^{n+4})$
iv. $f(n) = O(2^{4n})$

(iv), (i)

(i) $4^{2n} \approx 2^{4n+2}$

(iv) $2^{4N} \approx 2^{4n+2}$

2

↑ These are True

(b) Derive a recurrence relation for the running time T(n) of Mystery. Do not solve the recurrence.

```
FUNCTION Mystery(A[1..n]):
  IF n ≤ 1: - O(1)
    Return 0 - O(1)

  m = floor(n/3) - O(1)
  L = A[1..m] - O(1)
  R = A[m+1..n] - O(1)

  leftResult  = Mystery(L) - T(N/3)
  rightResult = Mystery(R) - T(2N/3)

  cross = 0 - O(1)
  for i from 1 TO m: - N/3
    for j from 1 TO n - m: - 2N/3
O(1) -  If (L[i] + R[j]) MOD 7 == 0:
        cross += 1 - O(1)

  Retrun leftResult + rightResult + cross - O(1)
```

~~$T(N) = 2T(N-m) + O(1)$~~

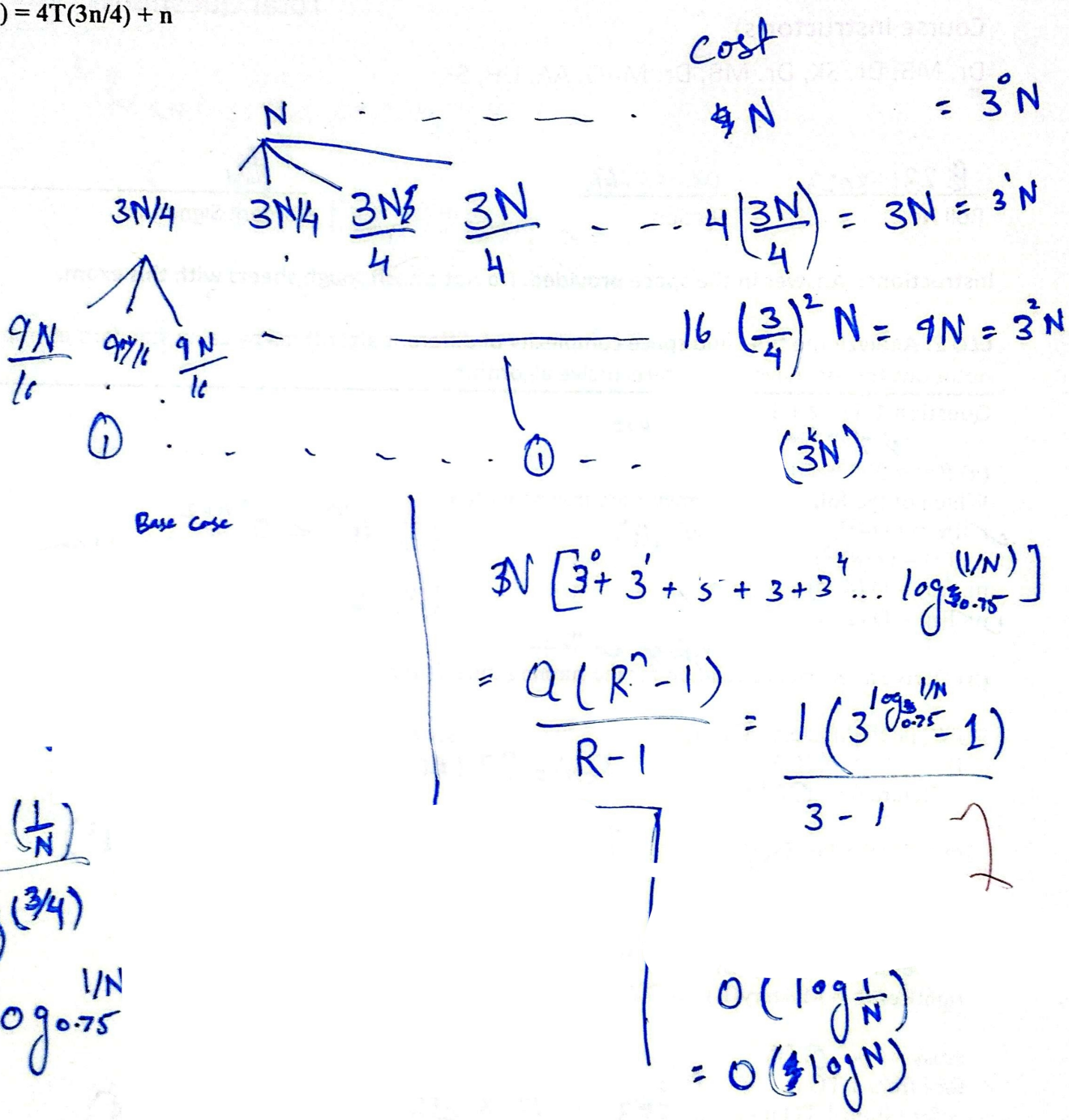
$T(N) = T(N/3) + T(2N/3) + O(N^2)$

2

$\frac{N}{3} \times \frac{2N}{3} = \frac{2N^2}{9} \approx O(N^2)$

(c) Apply the recursion tree technique to solve the given recurrence and state the asymptotic time complexity of the solution.

$T(n) = 4T(3n/4) + n$



CLO #1: Design algorithms using different algorithms design techniques i.e. Brute Force, Divide and Conquer, Dynamic Programming, Greedy Algorithms and apply them to solve problems in the domain of the program

Question 2: [2 Marks] Quick Sort's efficiency depends heavily on how the pivot is chosen. Describe a specific input and pivot-selection strategy that causes Quick Sort to run in $\Theta(n^2)$ time.

when the input is already sorted and pivot = n/2 where n is NO of inputs.

2

Question 3: [2+4 = 6 Marks] Recall the problem of finding the number of inversions. We are given a sequence of n numbers $a_1, \ldots, a_n$, which we assume are all distinct, and we define an inversion to be a pair $i < j$ such that $a_i > a_j$. We motivated the problem of counting inversions as a good measure of the sortedness of an array. However, one might feel that this measure is too sensitive. Let's call a pair a significant inversion **if $i < j$ and $a_i > 2a_j$**.

(a) Give an $O(n^2)$ algorithm to count the number of significant inversions between two orderings.[2 Marks]

```
int count=0;
for(int i=0; i<n-1; i++)
  {
    for(int j= i+1; j<n ; j++)
      { if (A[i]>2A[j])
          count++
      }
  }
return count;
```

2

(b) Give an O(n log n) algorithm to count the number of significant inversions between two orderings.[4 Marks]