National University of Computer and Emerging Sciences
Karachi Campus

Design and Analysis of Algorithms (CS2009)

Mid-1 Exam

Date: Sep 23rd 2025

Course Instructor(s)

Dr. Nasir Uddin, Dr. Kamran, Dr. Fahad Sherwani, Ms. Fizza Mansoor, Mr. Sandesh, Mr. Muhammad Kashif, Mr. Zulfiqar Ali

Total Time (Hrs): 1
Total Marks: 12.5
Total Questions: 3

_____________ ____________ _________________________
Roll No Section Student Signature

Do not write below this line

Attempt all the questions.

CLO #1: To apply acquired knowledge to solve computing problems complexities and proofs

---

Q1 (a): Suppose an algorithm has the recurrence relation **T(n)=** $\boldsymbol{aT\left(\frac{n}{b}\right) + f(n)}$. [1 mark]

a) What does increasing the constant "**a**" do to the growth rate?
b) What does increasing the constant "**b**" do?

Give a logical argument without solving the recurrence.

Solution:

Solution

let's take the exaple of Merge sort

Merge sort recurrence $T(n) = 2T\left(\frac{n}{2}\right) + n$

$a = 2$: it creates 2 subproblems.

$b = 2$: each subproblem is half the size

· $f(n) = n$: merging the two sorted halves

$T(n) = \Theta(n \log n)$

Increasing a: let suppose $a = 3$

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

merge sort split the array into 3 part instead there are more recursive calls at each level and recursion tree gets wider. increase total algorithm becomes slower.

Increasing b :- suppose Merge sort splits into 2 subpro[blems] of size $n/4$ instead of $n/2$.

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

Each subproblem is now much smaller.

[illegible] b will decrease total work [illegible]

Q1 (b): Solve the following recurrence relations by using the Master Theorem [2 marks]

a) $T(n) = 7T\left(\frac{n}{3}\right) + n^{\frac{1}{2}}$

b) $T(n) = 8T\left(\frac{n}{2}\right) + n^3 \log(n) + n^4 + 6$

Solution a:

Solution b:

Solution

$$T(n) = aT\left(\frac{n}{b}\right) + f(n).$$

$a = 7$

$b = 3$

$f(n) = n^{0.5}$, $d = 0.5$, $b^d \Rightarrow 3^{0.5} = \sqrt{3} = 1.732$

$7 > 1.732$

if $a > b^d$ Case 3 of Master Theorem

$\Theta(n^{\log_b a})$ if $a > b^d$

$\log_b a \Rightarrow \log_3 7 = 1.77$

therefore $\Theta(n^{1.77})$

$T(n) = \Theta(n^{1.77})$

Soution b:

$T(n) = 8T\left(\frac{n}{2}\right) + n^3\log(n) + n^4 +$

$f(n) = n^3\log(n) + n^4 + 6$

$= \Theta(n^4)$

$\therefore$ $d = 4$

$a = 8$

$b = 2$

$b^d = 2^4 = 16$

$\therefore$ $a < b^d$

$\therefore$ $T(n) = \Theta(n^d)$

$= \Theta(n^4)$

Q1 (c): Solve the following recurrence relations by using the Substitution Method. First find the exact solution and then write it in big O notations. [2 marks]

a) $T(n) = 16T\left(\frac{n}{2}\right) + 3n$

b) $T(n) = 8T\left(\frac{n}{2}\right) + \frac{n}{3}$

Solution a:

$T(n) = 16T\left(\frac{n}{2}\right) + 3n$

$T(n) = 16\left(16T\left(\frac{n}{2^2}\right) + 3\left(\frac{n}{2}\right)\right) + 3n$

$= 16^2 T\left(\frac{n}{2^2}\right) + 16\left(\frac{3}{2}\right)n + 3n$

$= 16^2\left(16T\left(\frac{n}{2^3}\right) + 3\left(\frac{n}{2^2}\right)\right) + 16\left(\frac{3}{2}\right)n + 3n$

$= 16^3 T\left(\frac{n}{2^3}\right) + 16^2\left(\frac{3}{2^2}\right)n + \left(\frac{16}{2}\right)(3)n + 3n$

$\vdots$

$T(n) = 16^i T\left(\frac{n}{2^i}\right) + \left(\frac{16}{2}\right)^{i-1}(3)(n) + \cdots \left(\frac{16}{2}\right)^2(3)n + \left(\frac{16}{2}\right)(3n) + 3n$

For base case $\frac{n}{2^i} = 1$

$\Rightarrow i = \log_2 n$

$\therefore$

$T(n) = 16^{\log_2 n} + \left[\left(\frac{16}{2}\right)^{i-1} 3n \ldots 8^2(3n) + 8(3n) + 3n\right]$

$= \left(2^{\log_2 n}\right)^4 + \left[8^{\log_2 n - 1} + \ldots 8^2 + 8 + 1\right]3n$

$= n^4 + 3n\left(\frac{8^{\log_2 n} - 1}{8 - 1}\right)$

$= n^4 + \frac{3}{7}n(n^3 - 1) = n^4 + \frac{3}{7}n^4 - \frac{3}{7}n$

$= \frac{10}{7}n^4 - \frac{3}{7}n$

$= \Theta(n^4)$

Solution b:

$$T(n) = 8T\left(\frac{n}{2}\right) + \frac{n}{3}$$
$$= 8\left(8T\left(\frac{n}{2^2}\right) + \frac{n}{2*3}\right) + \frac{n}{3}$$
$$= 8^2 T\left(\frac{n}{2^2}\right) + \frac{n}{2*3} + \frac{n}{3}$$
$$= 8^2\left(8T\left(\frac{n}{2^3}\right) + \frac{n}{2^2*3}\right) + \frac{n}{2*3} + \frac{n}{3}$$
$$= 8^3 T\left(\frac{n}{2^3}\right) + \frac{n}{2^2*3} + \frac{n}{2*3} + \frac{n}{3}$$
$$\vdots$$
$$= 8^i T\left(\frac{n}{2^i}\right) + \frac{n}{2^{i-1}*3} + \frac{n}{2^{i-2}*3} + \cdots \frac{n}{3}$$

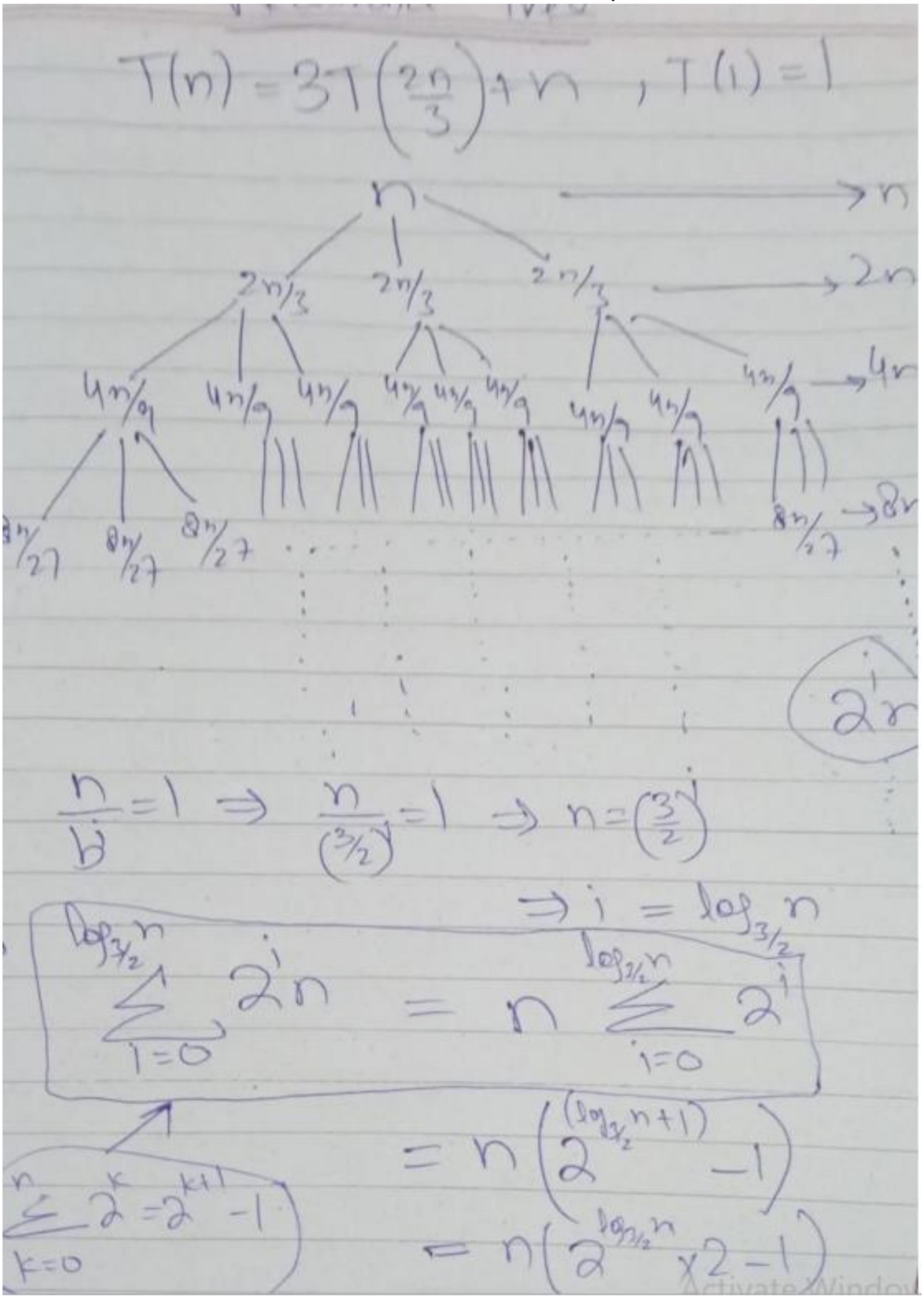
Assume

$$T(1) = 1 \Rightarrow \frac{n}{2^i} = 1 \Rightarrow n = 2^i \Rightarrow i = \log_2 n$$

$$\therefore \quad T(n) = 8^{\log_2 n} T(1) + \frac{n}{2^{\log_2 n - 1}*3} + \cdots \frac{n}{2*3} + \frac{n}{3}$$
$$= (2^3)^{\log_2 n} + \frac{n}{3}\left(\frac{1}{2^{\log_2 n - 1}} + \cdots \frac{1}{2} + 1\right)$$
$$= (2^{\log_2 n})^3 + \frac{n}{3}\left(\frac{1 - \left(\frac{1}{2}\right)^{\log_2 n}}{1 - \frac{1}{2}}\right)$$
$$= n^3 + \frac{2n}{3}\left(1 - \frac{1}{n}\right)$$
$$T(n) = \Theta(n^3)$$

Q1 (d): Solve the following recurrence relations by using the Recursion Tree Method. Assume $T(1) = 1$ [2 marks]

a) $T(n) = 3T\left(\frac{2n}{3}\right) + n$
b) $T(n) = T\left(\frac{n}{4}\right) + T\left(\frac{n}{2}\right) + n^2$
c) **Solution a:**



Solution b:

Recurrence tree

cn2

/ \

T(n/4) T(n/2)

If we further break down the expression T(n/4) and T(n/2),

we get following recursion tree.
cn2
/ \
c(n2)/16 c(n2)/4
/ \ / \
T(n/16) T(n/8) T(n/8) T(n/4)
Breaking down further gives us following
cn2
/ \
c(n2)/16 c(n2)/4
/ \ / \
c(n2)/256 c(n2)/64 c(n2)/64 c(n2)/16
/ \ / \ / \ / \
To know the value of T(n), we need to calculate sum of tree nodes level by level. If we sum the above tree level by level, we get the following series
T(n) = c(n^2 + 5(n^2)/16 + 25(n^2)/256) + ...
The above series is geometrical progression with ratio 5/16.
To get an upper bound, we can sum the infinite series.
We get the sum as (n2)/(1 - 5/16) which is O(n2)

CLO #2: To analyze complexities of different algorithms using asymptotic notations, complexity classes and standard complexity function

Q2 : Prove the following statements. [2 marks]

a) $log_a(n) = \theta(\log_b(n))$
b) $n^2 \log(n) - 10\log(n) + 16 = \Omega(n^2 \log(n)), also\ find\ c\ and\ n_o$

Solution a:

Let $f(n)$ be an arbitrary function such that $f(n) \in \Theta(\log_a n)$. Then by the limit rule, we know that

$$\lim_{n\to\infty} \frac{f(n)}{\log_a n} = C$$

for some constant C such that $0 < C < \infty$. We also have

$$\lim_{n\to\infty} \frac{\log_a n}{\log_b n} = \lim_{n\to\infty} \frac{\log_a n}{\log_a n / \log_a b} = \lim_{n\to\infty} \log_a b = \log_a b,$$

and $0 < \log_a b < \infty$. Hence we have

$$\lim_{n\to\infty} \frac{f(n)}{\log_b n} = \lim_{n\to\infty} \frac{f(n)}{\log_a n} \cdot \frac{\log_a n}{\log_b n} = \left(\lim_{n\to\infty} \frac{f(n)}{\log_a n}\right)\left(\lim_{n\to\infty} \frac{\log_a n}{\log_b n}\right) = C \cdot \log_a b,$$

and $0 < C \cdot \log_a b < \infty$. Hence by the limit rule, $f(n) \in \Theta(log_b n)$. A similar argument can be used to show that for an arbitrary $f(n) \in \Theta(log_b n)$, it must be the case that $f(n) \in \Theta(log_a n)$.

Solution b:

$$n^2\log(n) - 10\log(n) + 16 = \Omega(n^2\log(n))$$

$$n^2\log(n) - 10\log(n) + 16 \geq cn^2\log(n)$$

$$n^2\log(n) - 10\log(n) + 16 > n^2\log(n) - 10\log(n)$$

lets show

$$n^2\log(n) - 10\log(n) \geq cn^2\log(n)$$

$$\frac{n^2\cancel{\log(n)}}{n^2\cancel{\log(n)}} - \frac{10\cancel{\log(n)}}{n^2\cancel{\log(n)}} \geq \cancel{\frac{cn^2\log(n)}{n^2\log(n)}}$$

$$1 - \frac{10}{n^2} \geq c$$

For c to be positive $n \geq 4$

For $n = 4$

$$c \leq 1 - \frac{10}{4^2} = \frac{16-10}{16} = \frac{6}{16} = \frac{3}{8}$$

Hence $c = \frac{3}{8}$, $n_0 = 4$

$$n^2\log(n) - 10\log(n) + 16 = \Omega(n^2\log(n))$$

CLO #3: To construct and analyze real world problems solutions using different algorithms design techniques i.e. Brute Force, Divide and Conquer, Dynamic Programming, Greedy Algorithms.

Q3: Imagine you are analyzing data from a weather station placed on a long mountain range. The station records the altitude every kilometer along a single straight path. The data is stored in an array $\boldsymbol{altitude}\ [\mathbf{0} \ldots \ldots \boldsymbol{n}-\mathbf{1}]$. A peak is defined as a location $\boldsymbol{i}$ where the altitude is greater than or equal to its immediate neighbors. For locations at the ends of the range, we only need to compare them with their one existing neighbor.

Your task is to design an efficient algorithm to find the highest peak (the peak with the maximum altitude value) in the array using a Divide and Conquer approach. For instance, In the array $[\mathbf{10}, \mathbf{8}, \mathbf{5}, \mathbf{12}, \mathbf{7}]$, the peaks are the values $\mathbf{10}$ and $\mathbf{12}$ and $\mathbf{12}$ is also the highest peak.

You are also required to write pseudocode, write recurrence relation and find its time complexity.

[3.5 marks]

Solution:

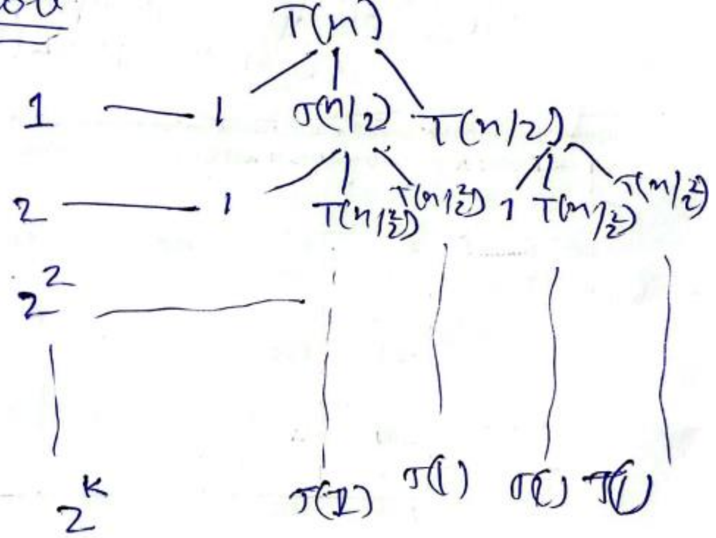
```
function find_highest_peak(A, low, high):
    // Base case: single element
    if low == high:
        return low

    // Base case: two elements
    if high == low + 1:
        if A[low] >= A[high]:
            return low
        else:
            return high

    // Divide: find middle point
    mid = (low + high) // 2)

    // find highest peaks in left and right halves
    left_peaks = find_highest_peak(A, low, mid)
    right_peaks = find_highest_peak(A, mid + 1, high)

    // Combine: return the higher peak
    if A[left_peak] >= A[right_peak]:
        return left_peak
    else:
        return right_peak
```

Recurrence Relation : **T(n) = 2T(n/2) + 1**

Time Complexity:

Recurrence Relation = $2T(n/2)+1$

Tree Method.



$1+2+2^2+2^3 + \ldots + 2^k$ (Geometric series $a=1, r=2$)

$= (a(r^k-1))/r-1 \Rightarrow 1(2^{\log n}-1) \Rightarrow 2^{\log n}$

using Logarithmic property $2^{\log n} = n$

$= O(n)$.