Rohan Nyati  50007 5940  R177219148  Batch-5 (AI & ML)

# DAA - Assignment

**Q.1** 1.) $T(n) = 2T(\sqrt{n}) + \log n$

Let $n = \log n$, i.e. $n = 2^m$.
Then $T(2^m) = 2T(2^{m/2}) + m$
Now, let $S(m) = T(2^m)$. Then $S(m) = 2S(m/2) + m$
This recurrence has the sol$^n$: $S(m) = \Theta(m \log m)$
So; $T(n) = T(2^m) = S(m) = \Theta(n \log m)$

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

2.) $T(n) = 3T(n/2) + n^2$

Using Master's Theorem; Here
$a = 3$, $K = 2$, $P = 0$
$n/b = n/2$ ; $\therefore b = 2$
$f(n) = n^2$
$\log_b^a = \log_2^3 \approx 1.58 < 2$
i.e. $f(n) < n^{\log_b^{a+\epsilon}}$, where $\epsilon$ is a constant
Case-3 implies here, Thus
$\therefore T(n) = \Theta(n^2)$

3.) $T(n) = 16T(n/4) + n$
$a = 16$, $b = 4$, $K = 1$, $P = 0$, $f(n) = n$
$\log_b^a = 2$
$\therefore \log_b^a > K$

$\therefore T(n) = O(n^2)$

Rohan Nyati
R177219148

4) $T(n) = 8\,T(n/4) - n^2$

$a = 8, b = 4, k = 2, P = 0, f(n) = n^2$

$\log_b a = \log_4 8 \approx 1.5$

$\therefore \log_b a < k$

$\therefore T(n) = O(n^2 \log n)$

5) $T(n) = 2\,T(n/2) + n \log n$

$a = 2, b = 2, f(n) = n \log n, k = 1, P = 1$

$\log_b a = 1 = k$

$\therefore T(n) = \Theta(n \log^2 n)$

6) $T(n) = 2T(n/2) + \frac{n}{\log n}$

$a = 2, b = 2, f(n) = n/\log n, k = 1, P = \frac{1}{\log n} = \log^{-1} n$

$\therefore P = -1$

$\log_b a = 1$

$\therefore \log_b a = k$

$\therefore O(n \log \log n)$

8) $T(n) = 0.5\,T(n/2) + 1/n$

$a = 0.5, b = 2, f(n) = n^{-1}, k = -1, P = 0$

$\log_b a = -1, \quad \therefore O(n^{-1} \log n)$

10) $T(n) = 2\,T(n/3) + n$

$a = 2, b = 3, f(n) = n, k = 1, P = 0$

$\log_3 2 = 0.6 < k$

$\hookrightarrow \therefore O(n \log n)$

Rohan Nyati
148

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

$a = 3, b = 4, f(n) = n, k = 1, P = 0$

$\log_b^a = \log_4^3 = 1.2 > k$

$\therefore O(n^{1.2})$

9.) $T(n) = 2T(\sqrt{n}) + 1$

Let $n = 2^m$ ; $T(2^m) = T(2^{m/2}) + 1$

Let $T(2^m) = S(m)$ ; $S(m) = 2S(m/2) + 1$

$\therefore S(m) = \Theta(m) = O(\log n)$

$\therefore T(n) = T(2^m) = S(m) = \Theta(\log n)$

6.) $T(n) = T(\sqrt{n}) + 1$

$T(n^{1/2}) = T(n^{1/2^2}) + 1$

$T(n) = T(n^{1/2^k}) + 1$

Assume $n = 2^m$

$T(2^m) = T(2^{m/2^k}) + 1$

Assume $T(2^{\frac{m}{2^k}}) = T(2)$

$\frac{m}{2^k} = 1 \Rightarrow k = \log_2^m$

Since $n = 2^m$ and $m = \log_2^n$

$k = \log \log_2^n$

$\boxed{T(n) = \Theta(\log \log n)}$

8.2 (E.) $T(n) = 2T(n-1) + 1$

The subproblem size for a node at depth $i$ is $n-i$.
Thus, the tree has $n$ levels and $2^{n-1}$ leaves.
The total cost over all nodes at depth $i$, for $i = 0, 1, 2, \ldots, n-1$ is $2^i$

$$T(n) = \sum_{i=0}^{n-1} 2^i = \frac{2^n - 1}{2-1} = 2^n - 1 = \Theta(2^n)$$

We guess, $T(n) \leq c2^n + n$ (Substitution Method)

$$T(n) \leq 2 \cdot c2^{(n-1)} + (n-1) + 1$$
$$= c2^n + n$$
$$= O(2^n)$$

(F.) $T(n) = 4T(n/2) + cn$

The subproblem size for a node at depth $i$ is $n/2^i$
Thus the tree has $\log n + 1$ levels and
$\therefore 4^{\log n} = n^{\log 4} = n^2$ levels.
The total cost over all nodes at depth $i$, for $i = 0, 1, 2, \ldots, \log n - 1$; is $4^i (cn/2^i) = 2^i cn$

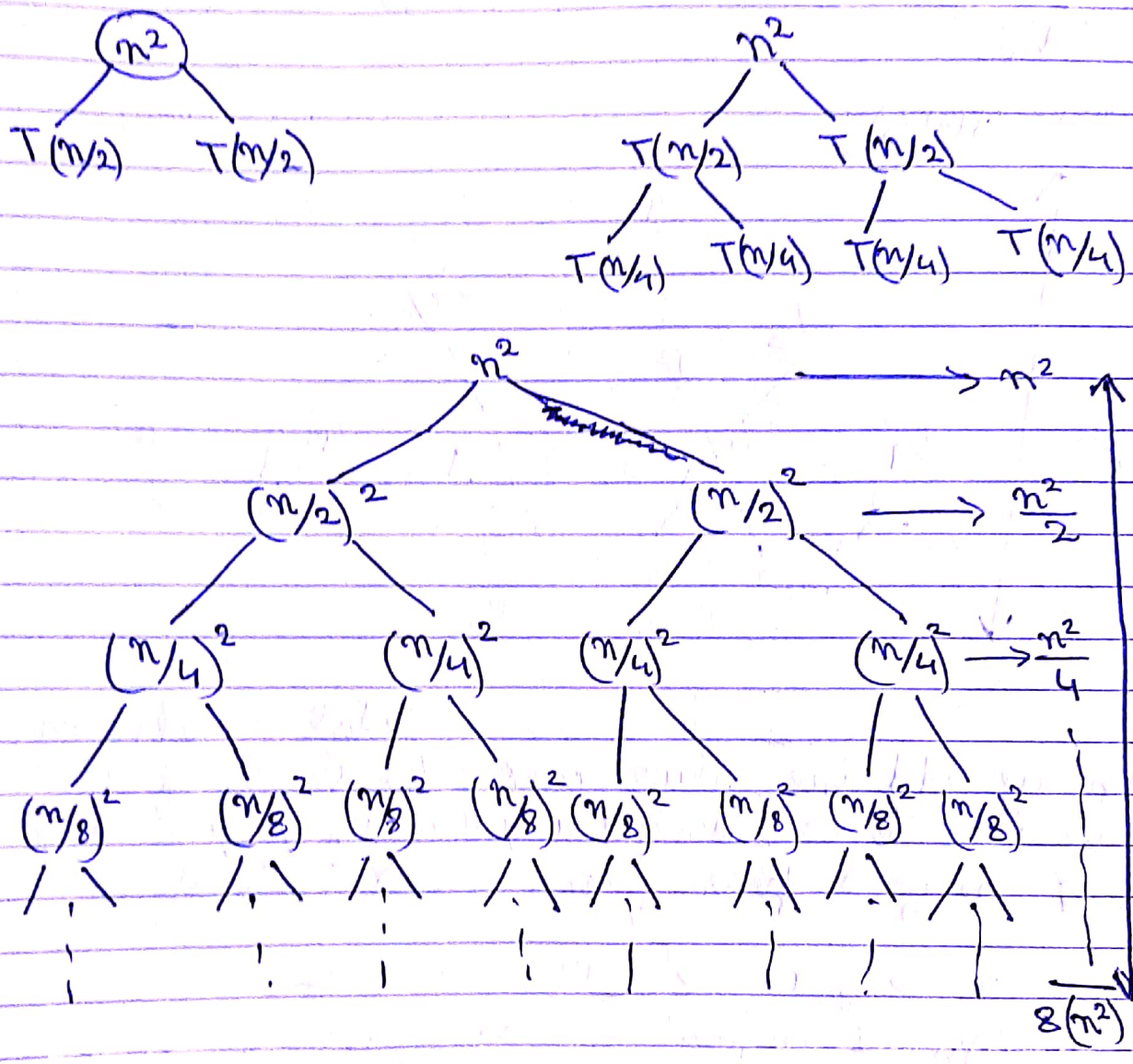
$$T(n) = \sum_{i=0}^{\log n - 1} 2^i cn + \Theta(n^2)$$

$$= \frac{2^{\log n} - 1}{2-1} + \Theta(n^2)$$

$$\boxed{T(n) = \Theta(n^2)}$$

Q.2] e (D) $T(n) = 2T(n/2) + n^2$

The recursion tree for the above recurrence is;




$$T(n) = n^2 + \frac{n^2}{2} + \frac{n^2}{4} + \cdots \text{ } \log n \text{ times}$$

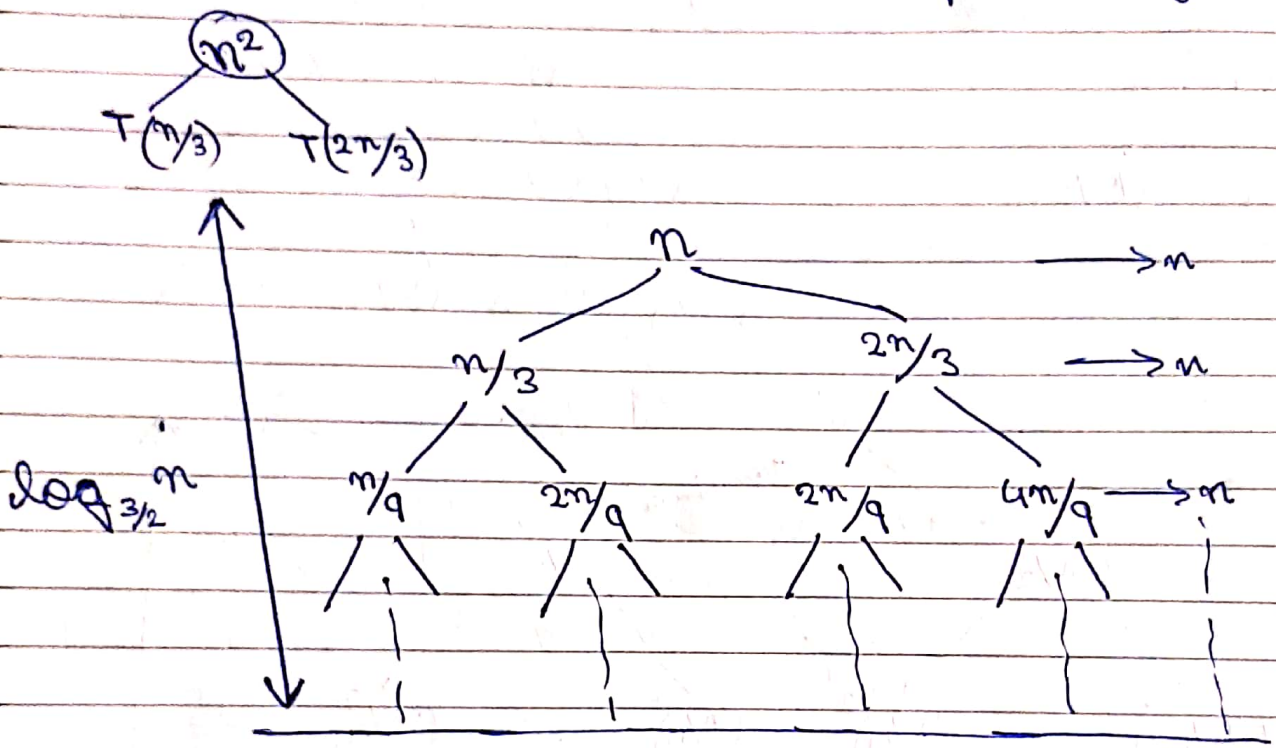
$$\leq n^2 \sum_{i=0}^{\infty} \left(\frac{1}{2^i}\right)$$

$$\leq n^2 \left(\frac{1}{1-\frac{1}{2}}\right) \leq 2n^2$$

$$\therefore \boxed{T(n) = \Theta(n^2)}$$

Q.2 (c.) $T(n) = T(n/3) + T(2n/3) + cn$

The given recurrence has the following recursion tree




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

When we add the values across the levels of the recursion tree, we get a value of $n$ for every level. The longest path from root to leaf is:

$$n \to \frac{2}{3}n \to \left(\frac{2}{3}\right)n \to \cdots 1$$

Since $\left(\frac{2}{3}\right)n = 1$ when $i = \log_{\frac{3}{2}} n$

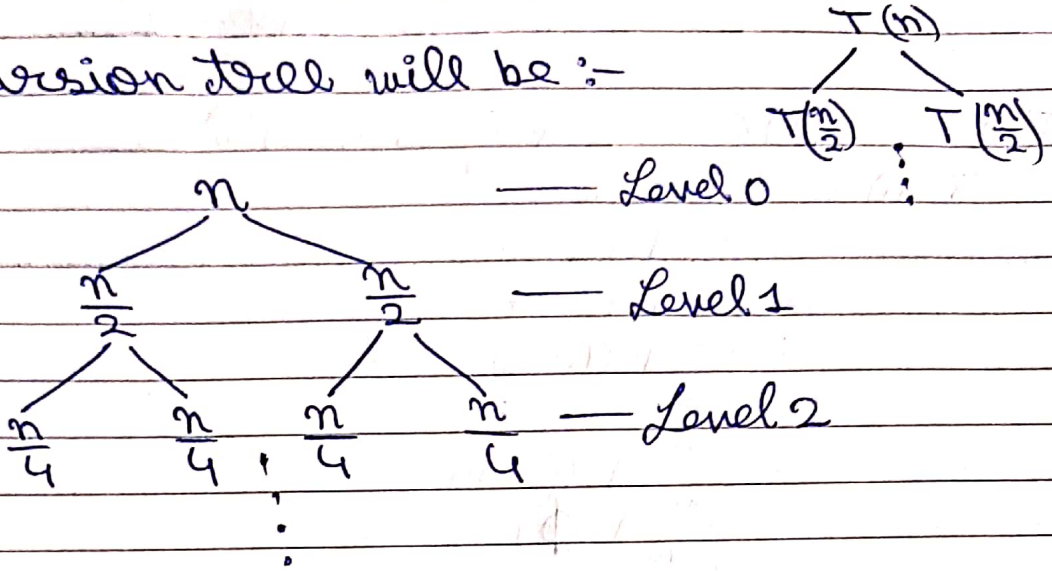
Thus the height of the tree is

$T(n) = n + n + \cdots + \log_{\frac{3}{2}} n$ times

$$\therefore \boxed{T(n) = \Theta(n \log n)}$$

Q.2] (A.) $T(n) = 2T(n/2) + cn$

The Recursion tree will be :-




Size of sub-problem at level-i = $n/2^i$

∴ Total no. of levels in the recursion tree = $\log_2 n + 1$

Level - $\log_2 n$ has $2^{\log_2 n}$ nodes i.e. n nodes.

∴ Cost of last level = $n \times T(1) = \Theta(n)$

Add costs of all levels of recursion tree and simplify the expression so obtained in terms of asymptotic notation

$$T(n) = (n + n + \cdots) + \Theta(n)$$

$$T(n) = n \times \log_2 n + \Theta(n)$$
$$= n \log_2 n + \Theta(n)$$

$$\therefore \boxed{T(n) = \Theta(n \log_2 n)}$$

Q.3 (A) $T(n) = \begin{cases} 3T(n-1) & \text{, if } n>0 \\ 1 & \text{, otherwise} \end{cases}$

Let us solve using Substitution

$T(n) = 3T(n-1)$
$= 3(3T(n-2))$
$= 3^2 T(n-2)$
$= 3^3 T(n-3)$
$\vdots$
$= 3^n T(n-n)$
$= 3^n T(0)$
$= 3^n$

∴ The Complexity of this func$^n$ is $3^n$.

(B.) $T(n) = \begin{cases} 2T(n-1) & \text{, if } n>0 \\ 1 & \text{, otherwise} \end{cases}$

Using Substitution Method

$T(n) = 2T(n-1)$
$= 2^2 T(n-2)$
$= 2^3 T(n-3)$
$\vdots$
$= 2^n T(n-n)$
$= 2^n T(0)$
$\because T(0) = 1$
$\therefore T(n) = O(2^n)$

∴ The Complexity of this func$^n$ is $2^n$.

Rohan Nyati



Q.4 (A) $f(n) = 27n^3 + 16n + 25$

Big oh (O) = $O(n^3)$

big omega = $\Omega(n^3)$ ; $n^3 \leq f(x) \leq 68n^3$

theta notation = $\Theta(n^3)$

(B.) $f(n) = n^4 + 100n^2 + 50$

→ $O(n^4), \Omega(n^4), \Theta(n^3)$ ; $n^4 \leq f(x) \leq 8n^4$

(C.) $f(n) = 2n^3 - 2n^2$

→ $O(n^3), \Omega(n^3), \Theta(n^3)$ ; $n^3 \leq f(x) \leq 5n^3$

(D.) $f(n) = n^2 + 1$

→ $O(n^2), \Omega(n^2), \Theta(n^2)$ ; $n^2 \leq f(x) \leq 2n^2$

(E.) $f(n) = 100\, n^{3/2} + 10000000$

→ $O(n^{3/2}), \Omega(n^{3/2}), \Theta(n^{3/2})$

(F.) $f(n) = 10n^2 + 5 \times 2^n$

→ $O(2^n), \Omega(n^2)$ ; $n^2 \leq f(x) \leq 5 \cdot 2^n$

(G.) $f(n) = n^2 + n! - 2^n$

→ $O(n^n), \Omega(1)$ ; $1 \leq f(x) \leq n^n$

Q.5] 1.) So, $T(n) = n \cdot (n-i) \cdot (n-j)$

$= (n^2 - ni)(n-j)$

$= n^3 - n^2 i - n^2 j + nij$

$\therefore T(n) = O(n^3)$

2.) $T(n) = \left(\frac{n}{2}\right) \cdot \left(\frac{n}{2}\right) \cdot (\log n)$

$\therefore T(n) = O(n^2 \log n)$

3.) int i, j, k, count = 0

$\frac{n}{2}$ [ for (i = n/2 ; i <= n ; i++)
  $\log n$ [ for (j = 1 ; j <= n ; j = j * 2)
    $\log n$ [ for (k = 1 ; k <= n ; k = k * 2)
      count++ ; ]]]

$\therefore T(n) = \frac{n}{2} (\log n)(\log n)$

$\therefore T(n) = O(n \log^2 n)$

4.) $T(n) = n * n$

$\therefore \boxed{T(n) = O(n^2)}$

6.) void function (int n)
{
$\sqrt{n}$ [ int i ; count = 0 ; for (i = 1 ; i * i <= n ; i++)
    count++ ; ]
}

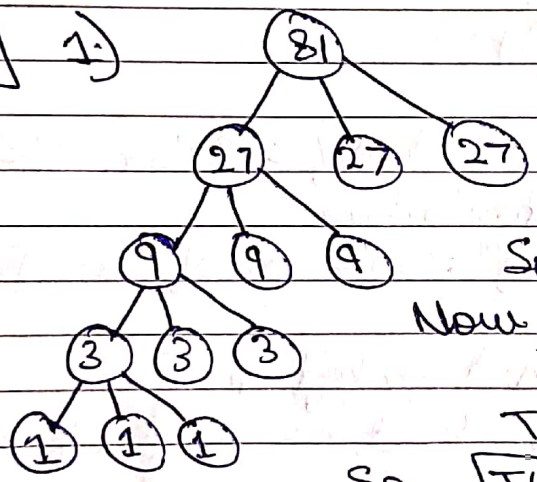
$\boxed{T(n) = O(\sqrt{n})}$

Rohan Nyati
148

5.) void function (int n)
{
int i=1, s=1; while(s<=n)
{
i++;
s=s+1; printf("*");
}
}

$\therefore$ $T(n) = O(n)$

Q.6 1.)




$T(n) = T(n/3) + 1$

$T(n/3) = T(n/3^2) + 1$

So, $T(n) = T(n/3^k) + k$

Now, let $\frac{n}{3^k} = 1$; $k = \log_3 n$

$T(n) = T(1) + \log_3 n$

So, $\boxed{T(n) = O(\log_3 n)}$

2.) (i) find the middle index
(ii) If mid is even, then compare arr(mid) and arr[mid+1].
(iii) If mid is odd, then compare arr[mid] and arr[mid-1].

$T(n) = T(n/2) + 2$
$T(n) = T(n/2^k) + 2n$

As $n = 2^k$; $k = \log_2 n$
$T(n) = T(1) + 2\log_2 n$

So, $\boxed{T(n) = O(\log_2 n)}$

Rohan Nyati
(148)

3.) Since the array is sorted we can use binary search method and div the divide the array from its mid element ; until we find the answer :-

$$T(n) = T(n/2) + 2$$
$$T(n) = T(n/2^k) + 2n$$

Assume $\frac{n}{2^k} = 1$ ; $k = \log_2 n$

$$T(n) = T(1) + 2\log_2 n$$

So , $T(n) = O(\log_2 n)$

4.) Since the array is sorted, we can use binary search, we check for

(i) If (arr[mid] < arr[mid-1] && arr[mid] < arr[mid+1])
a[mid]
- less than last or left half
- greater than last or right half

$$\boxed{T(n) = O(\log n)}$$

5.) (i) A variable x is created
(ii) write conditions for base cases. i.e. 0 or 1.
(iii) Run a loop until $x*x \leq n$ (n = given number)
for (x = 0; x*x ≤ n; n++)
Assume $x^2 > n$
Since $x = n$
$k^2 > n$
$k = \sqrt{n}$

$\therefore \boxed{T(n) = O(\sqrt{n})}$

Rohan Nyati
148

Q.10] Recursion Method to solve fibonnaci numbers.

```
int fib (int n)
{
    if (n <= 1)
        return n;
    return fib(n-1) + fib(n-2);
}
```

here,

$$T(n) = T(n-1) + T(n-2)$$
$$= O(2^{n-1}) + O(2^{n-2})$$

So, $\therefore T(n) = O(2^n)$