ASSIGNMENT - 01

Ans 1 Asymptotic notations are mathematical tools used to describe the limiting behaviour of a function as its input approaches infinity.

They help analyze the efficieny of algorithm by providing a concise way to express their time or space complexity.

1. Big O notation (O) : Represents the upper bound of an algorithm's running time in the worst case scenario.
   Example : $O(n^2)$ for a quadratic algorithm.

2. Omega notation (Ω): Represents the lower bound of an algorithm's running time in the best case scenario.
   Example : $\Omega(n)$ for a linear algorithm.

3. theta notation (Θ) : Represents both the upper and lower bounds providing a tight bound on the algorithm's running time.
   Example : $\Theta(n)$ for a linear algorithm.

ans 2:

Ans 2: The time complexity of the given code is $O(\log n)$. Since the variable "i" doubles in each iteration, the loop executes approximately $\log_2(n)$ times.

... bound up -

ans 3.

Ans 3. The recurrence relation $T(n) = 3T(n-1)$ represent the exponential growth. Therefore the time complexity is $O(3^n)$

Ans 4: The recurrence relation $T(n) = 2T(n-1) - 1$ represents exponential growth. Therefore the time complexity is $O(2^n)$

Ans 5: The time complexity of the given code is $O(n^{1/2})$ The loop iterates until the sum "s" exceeds "n" which happens approximately when "i" reaches $n^{1/2}$.

Ans 6: The time complexity of the given code is $O(n^{1/2})$. The loop iterates until $i*i$ is greater than n which happens approximately when i reaches $(n^{1/2})$.

Ans 7. The time complexity of the given codes is $O(n\log n)$ The outer loop runs $n/2$ to n times, the middle loop runs $\log n$ times & the inner loop also runs $\log n$ times. Therefore, the total time complexity is $O(n\log n)$

Ans 8. Inner loop runs n times and the outer loop runs n times, making it $O(n^2)$. Additionally. the function recursively calls with $n-3$ so the no. of times it recuses can be represented as $n/3$

$$O(n^2) * O(n/3) = O(n^3)$$

Ans 9. The outer loop runs n times and the inner

runs ~~ta~~ /i times. Therefore Time complexity of second function is $O(n) * O(n/1 + n/2 + n/3 \ldots n/n$

harmonic series is $\log n$

so Time complexity is $O(n \log n)$

Ans 10.

n^k growth rate increases polynomially with n.

c^n growth rate increases exponentially with n.

c^n grows faster than n^k

if $c > 1$ & $k > 0$ then

$$\lim_{n \to \infty} \frac{c^n}{n^k}$$

Therefore for any $c > 1$ & $k > 0$ c^n grows faster than n^k.

Ans 11. Time complexity for extracting minimum element from a heap using extractmin() operation typically has a complexity of $O(\log n)$.

n is number of elements in heap.

This complexity arises because after extracting the minimum element, the heap needs to restructure itself to ~~maint~~ maintain its properties, which involves heapify operations that take logarithmic time.

Ans 12 After deleting 15