Date.
Page No.

Name: Ankit Choudhary
Sec: DS I
Roll No: 18

Assignment - 01

1. Asymptotic notations are known as mathematical tools which are required to describe the limiting behaviour of a function as its input tends towards infinity.

(i) Big O notations (O): This notation represent the upper bound of an algorithm's running time in the worst case. For ex: $O(n^2)$ for a quadratic equation.

(ii) Omega notation: Represent the lower bound of an algorithm running time in the best case scenario. ex: $\Omega(n)$ for a linear algorithm.

(iii) Theta notation (Θ): Represent both the upper and lower bounds providing a tight bound on the algorithm running time.
ex: $\Theta(n)$ for a linear algorithm.

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.

Date.
Page No.

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

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

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 $(n^{1/2})$.

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

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 one runs log n time. Therefore, the total time complexity is $O(n \log n)$

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 reuses can be represent as n/3

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

Date.
Page No.

9. The outer and inner loop will run n times. Therefor time complexity of second function is

$O(n) * O(n/1 + n/2 + n/3 + \ldots\ldots n/n)$
harmonic series is $\log n$.
so time complexity is $O(n \log n)$.

10. n^k growth rate increase polynomially with n.
c^n growth rate increase 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.