Tutorial-1

Name :- Vikash Agarwal
Section :- B.Tech (IT)
Class Roll :- 32
University Roll :- 2015563

1. Asymptotic notations :- They are the notations or expression that are used to represent the complexity of an algorithm.

Types

① Theta (Θ) → Given the bound in which the function will fluctuate

② Big oh (O) :- $f(n) = O(g(n))$
$g(n)$ is tight upper bound of $f(n)$
i.e. $f(n)$ can never go beyond $g(n)$.

③ omega (Ω) :- $f(n) = \Omega(g(n))$
$g(n)$ is tight lower bound of $f(n)$.
i.e. $f(n)$ will never perform better than $g(n)$

2. Time complexity for

```
for (i=1 to) { i = i*2 }
```

$i = 1, 2, 4, 8 \ldots n$, $a = 1, r = 2$

$t_k = ar^{k-1}$

$n = \frac{r^k}{r} \Rightarrow rn = r^k$

Taking log on both side

$k \log_2 2 = \log_2(n) + \log_2(2)$

$k = \log_2(n) + 1$

$O(\log_2(n) + 1)$

$O(\log n)$

3) $T(n) = \{3T(n-1)$ if $n > 0$ otherwise $1\}$

using backward substitution

$T(n-1) = 3[3T(n-2)]$

$T(n-1) = 3^2[T(n-2)]$

$T(n-2) = 3^2[3T(n-2+1]$

$= 3^3 T(n-1)$

$\vdots$

$= 3^n(T(n-n))$

$= 3^n(T(0))$.

$T(0) = 1$

$T.C = O(3^n)$

4) $T(n) = \{2T(n-1)-1$ if $n > 0$ otherwise $1\}$

$T(n-1) = 2[2T(n-2)-1]-1$

$= 2^2(T(n-2))-2-1$

$T(n-2) = 2(2^2(T(n-3)-1))-2-1$

$T(n-2) = 2^3 T(n-3) - 4 - 2 - 1$

$T(n-3) = 2(2^3(T(n-4)-1)-4-2-1$

$= 2^4(T(n-4))-8-4-2-1$

$2^n(T(n-n)) - 2^{n-1} - 2^{n-2} \quad 2^0$

$T(0) = 1$

$2^n - 2^{n-1} - 2^{n-2} \quad 2^0$

$\Rightarrow 2^n - (2^n - 1)$

$T.C = O(1)$

5.
```
int S = 1, i=1
while (S<=n) {
    i++ ; S = S+i;
    printf("#");
}
```

$T.C = O(n)$

6. $O(\sqrt{n})$

7. loops

i	j	k
$n/2$	$\log n$	$\log n$

$T.C = \frac{n}{2} \times \log n \times \log n$

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

8. T.C $O(n^3)$.

9. $O(n^2)$.

10. Since polynomials grow slower than exponential n^k has an asymptotic upper bound of $O(a^n)$ for $a = 2$, $n_0 = 2$