Mahi Gupta.

Date
Page

Ans1 Asympotic notations is used to ~~maintain~~ describe the running time of an algorithm. i.e how much time an algorithm takes with a given input.

There tools are mathematical representation to represent the complexities

→ Big (O) notation.
It gives us upper bound. for a function $f(n)$ to write within a upper bound.

→ Omega notation (Ω)
Gives us lower bound for a function $f(n)$ to within a constant factor

→ Big theta (Θ)
It gives bound for a function $f(n)$ to within a constant factor

Ans2 for(i=1; i<=n ; i++)

Iter 1 : $i = 1 = 2^0$
2 $i = 2 = 2^1$
3 $i = 4 = 2^2$
4 $i = 8 = 2^3$
⋮

Iter p : $i = 2^{p-1} = n$

$2^{p-1} = n$

$p-1 = \log_2 n$

$p = \log n + 1$

$O(\log n)$

3. $T(n) = 3T(n-1)$ if $n > 0$ otherwise 1

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

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

$T(1) = 3T(0)$

$T(1) = 3$

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

$= 3T(1)$

$= 3 \times 3 = 9$

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

$= 3T(2)$

$= 3 \times 9$

$= 27$

$T(4) = 3T(4-1)$

$= 3T(3)$

$= 3 \times 27$

$= 81$

$T(n) = 3 + 9 + 27 + \cdots$

$T(n) \approx 3^n$

Hence time complexity

$O(n)$

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

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

$= 2T(0) - 1$

$= 2 \times 1 - 1 = 1$

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

$= 2T(1) - 1$

$= 2 \times 1 - 1 = 1$

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

$= 2T(2) - 1$

$= 2 \times 1 - 1 = 1$

⋮

$T(n) = 1$

$O(1)$.

5 Time complexity

iter 1 $S = 1 + 2$

iter 2 $S = 1 + 2 + 3$

iter 3 = $S = 1 + 2 + 3 + 4$

⋮

iter K = $\sum_{i=1}^{k} = n$

$\sum_{i=1}^{k} = \frac{K(K+1)}{2} \cdot O(K^2)$

$O(K^2) = n$

$O(\sqrt{2})$

6. iter 1 = $i=1$
iter 2 $i=2$
⋮
iter k = $\frac{n}{i}$

$k^2 = n$
$k = \sqrt{n}$
$O(\sqrt{n})$

7) Iter 1 $i = n/2 + 0$
iter 2 $i = n/2 + 1$
iter 3 $i = n/2 + 2$

iter k $\Rightarrow$ $i = n/2 + k - 1 = n$

$\frac{n}{2} + k - 1 = n$

$k = n - \frac{n}{2} + 1$

$= \frac{2n - n + 2}{n} = \frac{n+2}{2}$

$= \frac{n}{2} + 1$

$O(\frac{n}{2})$

iter 1 $j = 1 = 2^0$
iter 2 2 $j = 2 = 2^1$
3 $j = 4 = 2^2$
$j = 8 = 2^3$
⋮
iter k $\Rightarrow$ $j = 2^{k-1} = n$.
$k - 1 = \log n$

$K = \log n + 1$

$= \theta(\log n)$

iter 1 $K = 1$

iter 2 $K = 2$

iter 3 $K = 4$

iter 4 $K = 8$

$1 \times n \times \log n \times \log n$

$\theta(n(\log n)^2)$

iter $K = 2^{K-1} = n$

$2^{p-1} = n$

$p - 1 = \log n$

$p = \log n + 1$

$\theta(\log n)$

8 cannot be find.

Because we find the time complexity it must be algorithm as there is no terminating point so it is not a algorithm

9 iter 1 $\Rightarrow$ $i = 1 \rightarrow O(1)$

iter 1 $j = 1$

iter 2 $j = 2$ same for this also.