Name → Ayush Nautiyal

Roll No → 49

Sec → H

Tutorial - 1

Q1. What do you understand by Asymptotic notation. Define different Asymptotic notation with examples.

→ Asymptotic notation are the mathematical way of representing time complexity.

① Big O → Decides upper bound of an algorithm.

$f(n) \leq c.g(n)$ & $c > 0$, $n \geq k$, $k \geq 0$

Ex → $f(n) = 2n^2 + n$

$\therefore$ $2n^2 + n \leq 3n^2$, $c = 3$

② Big Ω → Decides lower bound of an algorithm.

$f(n) \geq c.g(n)$

Ex → $f(n) = 2n^2 + n$

$\therefore$ $2n^2 + n \geq 2n^2$, $c = 2$

③ Big θ → Bounds function from above and below bound.

$c_1.g(n) \leq f(n) \leq c_2.g(n)$

Ex → $f(n) = 2n^2 + n$

→ $2n^2 \leq 2n^2 + n \leq 3n^2$ $c_1 = 2$, $c_2 = 3$

Q2. what should be time complexity of

for (i=1 to n)

{ i = i * 2; }

→ Complexity → 1, 2, 4, 8, ... n

$\Rightarrow 2^0, 2^1, 2^2, 2^3, \dots 2^k$

$r = \frac{2}{1} = 2$, $a = 1$

$n = ar^{k-1} = 2^{k-1} = \frac{2^k}{2}$

$2n = 2^k$

$\log_2 n = \cancel{\log} k \log 2$

$k = \log_2 n$

∴ complexity will be $O(\log n)$

Q3. $T(n) = \{3T(n-1)$ if $n>0$, otherwise $1\}$

→ Complexity. $T(0) = 1$

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

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

$T(3) = 3T(2) = 3*3*3$

⋮

$T(n) = 3^n$

∴ complexity will be $O(3^n)$.

Q4: $T(n) = \{2T(n-1) - 1$ if $n > 0$, otherwise $1\}$

complexity → $T(0) = 1$

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

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

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

⋮

$T(n) = 1$

So complexity will be $O(1)$.

Q5. what should be time complexity of

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

→ complexity - $s = 1+2+3+4+\ldots n$

$s = \frac{n*(n+1)}{2} = n^2$

while $(s<=n)$ = $n^2 <= n$

$n <= \sqrt{n}$

∴ complexity will be $O(\sqrt{n})$.

Q6: Time complexity of

```
void function (int n)
{ int i, count = 0;
  for (i=1; i*i <= n; i++)
      count++;
}
```

complexity: $i = 1, 2, 3, \ldots k$

loop ends when $i * i <= n$

$k * k <= n$

$k^2 <= n$

$k \leq \sqrt{n}$

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

Q7: Time complexity of

```
void function (int n)
{  int i, j, k, count = 0;
   for (i = n/2, i <= n; i++)
   for (j = 1; j <= n; j = j*2)
   for (k = 1; k <= n; k = k*2)
       count++;
}
```

$\rightarrow$ $i = \frac{n+1}{2} = O(n)$

$j = O(\log n)$, $k = O(\log n)$

$\therefore d(i*j*k) = O(n * \log^2 n)$

Q8. fun (int n)

```
{ if (n==1)
    return;
  for (i=1 to n) {
  for (j=1 to n) {
   printf (" * ");
  } }
  fun (n-3);
}
```

complexity → $T(n) = T(n-3) + n^2$

$T(1) = 1$

→ $T(4) = T(4-3) + 4^2$

$= 1^2 + 4^2$

$T(7) = T(7-3) + 7^2$

$= 1^2 + 4^2 + 7^2$

$T(10) = T(10-3) + 10^2$

$= 1^2 + 4^2 + 7^2 + 10^2$

$$T(n) = \frac{n(n+1)(2n+1)}{6}$$

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

Q9. Time complexity of

void function (int n)

{ for (i=1 to n)

{ for (j=1; j<=n; j=j+i)

printf ("*"); }}

→ complexity → $n + \frac{n}{2} + \frac{n}{3} + \frac{n}{4} \cdots \frac{n}{n}$

$= n\left(\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots \frac{1}{n}\right)$

$= n(\log n)$

⇒ complexity will be $O(n \log n)$.

Q10. For the function n^k and c^n what is asymptotic relationship between these two functions?

Assume $k >= 1$ & $c > 1$ are constant. Find out value of c and n_0 for which relation holds.

→ $f(n) = n^k$, $f_2(n) = c^n$

Asymptotic relationship between f_1 and f_2 is Big O

i.e $f_1(n) = O(f_2(n)) = O(c^n)$

as $n^k \leq c_1 * c^n$ [c_1 is some constant].

—x—x——x—