10-3-22

DAA Lab

Jatin Tiwari
CST Spl 2
39

Date: / /

1. What do you understand by asymptotic notations. Define different asymptotic notations with examples.

Ans.1. Asymptotic notations — These are the mathe-matical relations used to describe the running time of an algorithm when the input tends towards a particular value or a limiting value.

The three important types of asymptotic relations are as follows:

① Big-O :

- This notation defines an upper bound of an algorithm.
- The function $f(n) = O(g(n))$ iff $f(n) <= c \cdot g(n)$ for $n >= n_0$, where c and n_0 are const.
- Here, $g(n)$ is known as upper bound on values of $f(n)$.
- For example- $f(n) = 3n + 3$, $g(n) = 4n$.

② Big-Ω :

- This notation provides an asymptotic lower bound.
- The function $f(n) = \Omega(g(n))$ if $f(n) >= c \cdot g(n)$ $\forall n >= n_0$, where c and n_0 are const.
- Here, $g(n)$ is known as lower bound on values of $f(n)$.
- For example- $f(n) = 3n + 2$ and $g(n) = 3n$.

Date: / /

③ Big-Θ:

- It bounds a fun^n from above and below, so it defines exact asymptotic behaviour. Hence, it's also known as tightly bound.
- function $f(n) = \Theta(g(n))$ if $c_1 \cdot g(n) <= f(n) <= c_2 \cdot g(n)$ $\forall$ $n >= n_0$ where c_1, c_2 and n_0 are constants.
- For example- $f(n) = 3n+2$, $g(n) = n$, $c_1 = 3$, $c_2 = 4$.

Q.2. What should be time complexity of:-

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

Time complexity = $O(n \log n)$.

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

Ans- $T(n) = 3T(n-1)$ — ①

Put $n = n-1$ in ①,

$T(n-1) = 3T(n-2)$ — ②

Put ② in ①,

$\Rightarrow$ $T(n) = 3(3T(n-2))$ — ③

Put $n = n-2$,

$T(n-2) = 3T(n-3)$ — ④

Put ④ in ③,

$T(n) = 3(3(3T(n-3)))$ — ⑤

In general, $T(n) = 3^k(T(n-k))$

Put $n-k = 0$,

$n = k$.

$\therefore$ $T(n) = 3^n(T(n-n)) = 3^n T(0) = 3^n$

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

Date: / /

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

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

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

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

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

$= 2^3 T(n-3) - 2^2 - 2^1 - 2^0$

$\vdots$

$= 2^n T(n-n) - 2^{n-1} - 2^{n-2} - 2^{n-3} \ldots 2^2, 2^1, 2^0$

$= 2^n - 2^{n-1} - 2^{n-2} - 2^{n-3} \ldots 2^2 - 2^1 - 2^0$

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

$\because \quad 2^{n-1} + 2^{n-2} + \ldots + 2^0 = 2^{n-1}$

$T(n) = 1$.

Time complexity = $O(1)$

5. Time complexity of:

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

Let loop execute n times. Now, the loop will execute as long as s is less than n

After 1st iteration:

$s = s + 1$

After 2nd iteration:

$s = s + 1 + 2$

And it goes for n interations,

$1 + 2 + \ldots + n <= n$.

$\Rightarrow (x^{(x+1)/2}) <= n.$

$\Rightarrow O(x^2) <= n.$

$\Rightarrow x = O(\sqrt{n})$

Time complexity = $O(\sqrt{n})$.

(6) Time complexity of:

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

Let 'k' be max pos. value such that

$k^2 \leq n.$

$k = \sqrt{n}.$

$i^2 \leq n.$

~~$i=1$~~

$\therefore \sum i = 1 + 1 + \ldots$ k times

$i=1$

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

7. Time complexity of:

```
void function (int n)
{
  int i, j, k, count=0;
```

Date: / /

```
for (i = n/2; i <= n; i++)
    for (j = 1; j <= n; j = j * 2)
        for (k = 1; k <= n; k = k * 2)
            count++;
}
```

for $k = k * 2$

$k = \underbrace{1, 2, 4, 8, \ldots, n}_{k}$

$a = 1, \; r = 2.$

G.P. $= k = \dfrac{a(r^n - 1)}{r - 1} = \dfrac{1(2^n - 1)}{}$

$n = 2^k$

$k = \log n.$

i	j	k
$n/2$	$\log n$	$\log n * \log n$
$(n+2)/2$	$\log n$	$\log n * \log n$
⋮	⋮	⋮
n	$\log n$	$\log n * \log(n)$

$n * \log n * \log n$

$O(n \log n)$ or $O((\log n)^2)$.

(8) Time complexity of

```
function (int n)
{
    if (n == 1) return;
```

Date: / /

```
for(i=1 to n)
    for(j=1 to n)
        printf("*");
    }
}
function(n-3);
}
```

for:- for(i=1 to n)
we get j=n times every time.
∵ $i \times j = n^2$.

Now, $T(n) = n^2 + T(n-3)$;
$T(n-3) = (n-3)^2 + T(n-6)$;
$T(n-6) = (n-6)^2 + T(n-9)$;

$T(1) = 1$;

Now, subs. each value in $T(n)$,
$T(n) = n^2 + (n-3)^2 + (n-6)^2 + \dots + 1$

Let
$(n-3k) = 1$
$k = (n-1)/3$.
total terms $= k+1$
$T(n) = n^2 + (n-3)^2 + (n-6)^2 + \dots + 1$.
$T(n) \simeq n^2 + n^2 + n^2 + \dots$ (k times +1)
$T(n) \simeq kn^2$
$T(n) \simeq \frac{(n-1)}{3} \times n^2$
$T(n) = O(n^3)$.

Date: / /

9. Time complexity of:

```
void function(int n)
{
    for (i=1 to n)
    {
        for (j=1; j<=n; j=j+1)
            printf("*");
    }
}
```

9. for:- i=1 j= 1+2+...(n ≥ j+i)

i=2 j= 1+3+5+... "

i=3 j=1+4+7+... "

nth form of AP is

$T(m) = a + d \times m$

$T(m) = 1 + d \times n$

$(n-1)/d = m$

for i=1 (n-1)/1 times

i=2 (n-1)/2 time

i=3 (n-1)/3 times

⋮

i=n-1 1

we get

$T(n) = i_1 j_1 + i_2 j_2 + \cdots + i_{n-1} j_{n-1}$

$= \frac{(n-1)}{1} + \frac{(n-2)}{2} + \frac{(n-3)}{3} + \cdots + 1$

$= n + n/2 + n/3 + \cdots + n/(n-1) - n \times 1 + \cdots$

$= n\left(1 + 1/2 + 1/3 + 1/(n-1)\right) - n + 1$

$= n(\log n) - n + 1.$

Date: / /

Since $\int 1/n = \log n$

$T(n) = O(n \log n)$.

(10) For the fun^n, n^k and c^n, what is the asymp. relation bet. them for functions? Assume that $k >= 1$ and $c > 1$ are constants. Find out the val. of c and n_0 for which rel^n holds.

as given n^k and c^n

relation between n^k and c^n is:

$n^k = O(c^n)$

$n^k \leq a(c^n)$

$\forall\ n \geq n_0$ and

const. $a > 0$.

for $n_0 = 1$

$c = 2$

$\Rightarrow 1^k \leq a^2$

$\Rightarrow n_0 = 1$ & $c = 2$.