DESIGN AND ANALYSIS OF ALGORITHM

Tutorial - 1

Name - DEEPTI PANDEY
SECTION - CST SPL1
CLASS ROLL NO - 21
UNIVERSITY ROLL NO - 2016724

① What do you understand by Asymptotic Notations. Define different Asymptotic notation with examples.

Asymptotic notations are the notations which are used to describe the running time of an algorithm when the input tends towards a particular or a limiting value.

Different asymptotic notations with examples:

(i) big O - This notation describes the worst-case running time of a program. It is determined by counting number of iterations an algorithm takes in worst-case scenario with an input of N. One example is $O(\log n)$ ~~dat~~ is the Big-O of a binary search.

(ii) Big-Ω - This tells the running time (best) of the program. It is calculated by counting number of iterations an algorithm takes in best-case scenario. One example of this is bubble sort which as a running time of $\Omega(N)$ in best case scenario.

(iii) Big Theta (Θ) - This is counted by counting number of iterations the algorithm always takes with an input of n.

② What should be the time complexity of

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

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

Time complexity for above given code is $O(n)$.

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

$T(n) = 3T(n-1)$ —— (1)

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

putting $n = n-1$ in (1),

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

$\Rightarrow T(n-1) = 3T(n-2)$ —— (2)

putting value of $T(n-1)$ from (2) in (1),

~~$T(n) = 3T[$~~ $T(n) = 3(3T(n-2))$ —— (3)

putting $n = n-2$ in eq (1),

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

$\Rightarrow T(n-2) = 3T(n-3)$ —— (4)

putting value of $T(n-2)$ in (3),

$T(n) = 3(3(3T(n-3))$ —— (5)

$T(n) = 9(3T(n-3))$

$T(n) = 27T(n-3)$

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

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

Given, $T(n) = 2T(n-1)-1$ —— ①

putting $n = n-1$ in ①,

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

$\Rightarrow T(n-1) = 2T(n-2)-1$ —— ②

putting $T(n-1)$ from ② to ①,

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

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

putting $n = n-2$ in ①,

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

$= 2T(n-3)-1$ —— ③

putting $T(n-2)$ in ①,

$T(n) = 4[2T(n-3)-1]-3$

$= 8T(n-3)-4-3 = 8T(n-3)-7$ —

putting $n = n-3$ in ①,

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

$= 2T(n-4)-1$ —— ④

putting ④ in ①,

$T(n) = 8[2T(n-4)-1]-7$

$= 16T(n-4)-8-7 = 16T(n-4)-15$

$\Rightarrow$ ~~$T(n) =$~~ $T(n) = 2^k T(n-k) - (2^k - 1)$

let $n-k = 1 \Rightarrow n = k$

Now,

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

Time Complexity : $O(2^n * C - 2^n + 1)$

$= O(2^n)$

⑤ What should be time complexity of -

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

i= 1, 2, 3, 4, 5, 6 ....

S= 1+ 3 + 6 + 10 + 15 .....

Sum of $S = 1+3+6+10+\ldots\ldots+n$ — ①

$S = 1+3+6+10+\ldots\ldots+n$ — ②

~~From ①~~ Subtracting ② from ①,

$0 = 1+2+3+4+\ldots\ldots n - T_n$

$T_K = 1+2+3+4+\ldots\ldots K$

$T_K = \frac{1}{2} K(K+1)$

for K iterations

$1+2+3+\ldots\ldots+K <= n$

$\Rightarrow \frac{K(K+1)}{2} <= n$

$\frac{K^2+K}{2} <= n \Rightarrow O(K^2) <= n$

$\Rightarrow K = O(\sqrt{n}) \Rightarrow T(n) = O(\sqrt{n})$

⑥ Time Complexity of

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

$i^2 <= n$

$\Rightarrow i <= \sqrt{n}$

$i = 1, 2, 3, 4, \ldots, \sqrt{n}$

$\sum_{i=1}^{n} 1+2+3+4+\ldots\ldots+\sqrt{n}$

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

$T(n) = \frac{n \times \sqrt{n}}{\cancel{2}}$

$\Rightarrow T(n) = O(n)$

⑦ Time complexity of

```
void fun (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++;
}
```

for $K = K * 2$

$K = 1, 2, 4, 8, \ldots\ldots, n$

G.P. $a = 1,\ r = 2$

$$\frac{a(r^n - 1)}{r - 1}$$

$$= \frac{1(2^K - 1)}{1} \Rightarrow n = 2^K$$

$$\cancel{\Rightarrow \log_2 n = \log_2 K}$$

$$\Rightarrow \log_2 n = K$$

$i \Rightarrow 1, 2, \ldots\ldots n$

$j \Rightarrow \log n, \log n, \ldots\ldots \log n$

$K \Rightarrow \log n * \log n, \log n * \log n \ldots\ldots \log n * \log n$

So, $O(n * \log n * \log n)$

$= O(n \log n)$

(8) Time Complexity of

```
function (int n)
{ int (n==1)
      return ;
  for (i=1 to n)
    { for (j=1 to n)
        { print ('*');
        }
    }
    function (n-3);
}
```

$T(n) = T(n/3) + n^2$

Here, $a=1$, $b=3$, $f(n)=n^2$

$c = \log_b a = \log_3 1 = 0$

$\Rightarrow n^c = n^0 = 1 > f(n)$

So, $T(n) = \Theta(n^2)$

(9) Time Complexity of

~~void~~ function (int n)

```
{ for(i=1 to n)
    { for (j=1 ; j<=n; j=j+i)
          print ("*");
    }
}
```

$j=1, \quad j = 1, 2, 3, 4, \ldots\ldots, n$

$i=2, \quad j = 1, 3, 5, \ldots\ldots, n$

$i=3, \quad j = 1, 4, 7, \ldots\ldots n$

for $i=n \Rightarrow j = 1 \ldots$

$$\sum_{j=n}^{1} n + \frac{n}{2} + \frac{n}{3} + \frac{n}{4} + \ldots\ldots + 1$$

$$\Rightarrow \sum_{j=n}^{1} n\left[1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \ldots\ldots + \frac{1}{n}\right]$$

$$\Rightarrow \sum_{j=n}^{1} n[\log n]$$

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

(10) for functions, ~~n^k &~~ n^k and c^n, what is the asymptotic relationship between these functions. Assume that $k >= 1$ and $c > 1$ are constants. Find out the value of c and n_0 for which relation holds.

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

As, $n^k \leq c^n$

for all $n \geq n_0$ some constant $a > 0$

for $n_0 = 1$
$c = 2$

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

$n_0 = 1$ and $c = 2$.