Design and Analysis of Algorithm

Tutorial 1

Name - Shikha , Section - C

University Roll No - 2014853

Q.1 : Asymptatic notations are languages that allows us to analyze an algorithm's running time by identifying its behaviour as the input size of algorithm.

Types :

i) Big O : It is commanly used for worst case, and gives us upper bound for the growth rate of runtime algorithm.

eg : Big O notation for linear search is O(n)

ii) Big Omega : It is notation used for best case complexity, it provides us with an asymptotics lower bound.

eg. Big Omega of linear search is $\Omega(1)$

iii) Theta : It is used for tight bound on the growth rate of runtime of an algorithm.

eg. Theta of linear search is $\Theta(n)$

iv) Small Θ : It is used to donate the upper bound (i.e not asyntotically tight).

$f(n) = O(g(n)) \quad \forall \quad f(n) < c(g(n))$ where $c > 0$

V) small omega : To denote lower bound that is asymptatically tight.

Q.2 :
```
for (i = 1 to n)
{
    i = i * 2
}
```
Time Complexity - $O(\log n)$

Q.3 : $T(n) = 3T(n-1)$

$T(1) = 1$

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

$T(3) = 3T(3-1) = 6T$

$T(4) = 3T(4-1) = 9T$

$\vdots$

$T(n) = (n-1)^3$ Time Complexity : $O(3^n)$

Q.4. $T(n) = 2(T(n-1)-1)$

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

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

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

$T(n) = 8T(n-3)-4-2-1$

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

$T(n) = 16T(n-4)-8-4-2-1$

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

Time Complexity : $O(1)$

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

s	i
1	1
3	2
6	3
10	4

Time Complexity : $O(\sqrt{n})$

Q.6. void function (int n)

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

$i * i = n$

$i^2 = n$

$i = \sqrt{n}$

Time complexity : $O(\sqrt{n})$

Q.7. void function(int n) {

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

Time Complexity - $O(n \log^2 n)$

Q.8. function (int n) {

```
int (n==1) return;
for (i=1 to n) {
 for (j=1 to n) {
      printf("*");
}  }
```

```
    function(n-3);
}
```

Time Complexity : $O(n)$

Q.9.
```
void function(int n)
{
  for(i=1 to n)
  {
    for(j=1; j<=n; j++)
      printf("*");
  }
}
```

Time Complexity : $O(n \log n)$

Q.10. n^k is $O(c^n)$ as for example

If we take $n=2$, $k=2$, $C=2$

Then $2^2 \leq 2^2$

So, C^n is upper limit of n^k.

Shikha