__/__/__

Design and Analysis of Algorithms

Name: PURU SINGH
Class: CST-SPL-1
Roll no: 60

Tutorial-1

Q1 What do you understand by asymptomatic notations. Define different asymptotic notations with examples.

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

There are mainly three asymptotic notations:

1) Big O notation : Gives the worst case time complexity

2) Omega notation (ω) : Gives the best case ~~time~~ complexity.

3) Theta notation (Θ) : Gives the average case complexity.

___/___/___

For eg: In bubble sort, when the input array is already sorted, the time taken by the algorithm is linear, i.e best case.
But, when the same array is in reverse order, the time taken is quadratic, i.e worst case.

Q2 for (i = 1 to n)
{
    i = i * 2;
}

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

$a_k = a r^{k-1}$

$a = 1 \Rightarrow r = 2$

$n = 2^{k-1}$

$k = \log_2 n + \cancel{1}$

$T(n) = O(\log_2 n)$ Ans

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

$T(n) = 3T(n-1)$ — (1), $T(0) = 1$ — (2)

put $n = n-1$ in eq (1)

$T(n-1) = 3T(n-2)$ — (3)

put (3) in (1)

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

$= 3^2 T(n-2)$ — (4)

put $n = n-2$ in eq (1)

$T(n-2) = 3T(n-2-1) = 3T(n-3)$ — (5)

putting (5) in (4)

$T(n) = 3^2[3T(n-3)] = 3^3 T(n-3)$

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

let $n-k=0$

$n=k$

$T(n) = 3^n T(0)$

as $T(0) = 1$ (from (2))

$T(n) = 3^n$

$T(n) = O(3^n)$ Ans

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

$T(n) = 2T(n-1)-1$ — (1), $T(0) = 1$ — (2)

let $n=n-1$ in eq (1)

$T(n-1) = 2T(n-2)-1$ — (3)

put (3) in (1)

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

$= 2^2 T(n-2) - 2 - 1$ — (4)

let $n=n-2$ in (1)

$T(n-2) = 2T(n-3)-1$ — (5)

put (5) in (4)

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

$= 2^3 T(n-3) - 2^2 - 2 - 1$ — (6)

generalizing

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

let $n-k=0$

$n=k$

$T(n) = 2^n T(n-n) - 2^{n-1} - 2^{n-2} - \dots - 2^0$

from (2) $T(n) = 2^n - 2^{n-1} - 2^{n-2} - \dots - 2^0$

$= 2^n - \dfrac{(1 \times (2^n-1))}{2-1} \Rightarrow T(n) = (2^n - 2^n + 1)$

$\Rightarrow T(n) = O(1)$

_/_/__

Q5 Time complexity of:-

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

for $i=1$, $s=1$

$i=2$, $s=1+2$

$i=3$, $s=1+2+3$

sum of n natural numbers

so, $s=\frac{k(k+1)}{2}$

To break out of loop, $s>n$

$$\frac{k(k+1)}{2} > n$$

$$\frac{k^2+k}{2} > n$$

$$k^2 > n$$

$\Rightarrow$ $O(k) = \sqrt{n}$ Ans

Q6 Time complexity of -

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

$i = 1, 2, 3, \ldots n$
$i^2 = 1, 4, 9 \ldots n$

so $i^2 <= n$ or $i <= \sqrt{n}$

Now, $a_k = a + (k-1)d$
$a = 1, d = 1$
$a_k <= \sqrt{n}$
$\sqrt{n} = 1 + (k-1).1$
$\sqrt{n} = k$
$T(n) = O(\sqrt{n})$ Ans

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++;
}
```

i	j	K
$n/2$	$(\log_2 n)$	$\log_2 n$
⋮	⋮	⋮
n		
$(\frac{n}{2}+1)$ times	$\log_2 n$	$\log_2 n$

$O(i * j * k) = O((\frac{n}{2}+1) \times \log_2 n \times \log_2 n)$

$= O((\frac{n}{2}+1) \times (\log_2 n)^2)$

removing constants,
$T(n) = O(n(\log_2 n)^2)$ Ans

Q8 Time complexity of -

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

$T(n) = T(n-3) + n^2$ — (1)

$T(1) = 1$ — (2)

let $n = n-3$ in (1)

$T(n-3) = T(n-3-3) + (n-3)^2$ — (3)

put (3) in (1)

$T(n) = T(n-6) + (n-3)^2 + n^2$ — (4)

put $n = n-6$ in (1)

$T(n-6) = T(n-3-6) + (n-6)^2$ — (5)

put (5) in (4)

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

⇒ Generalizing ⇒

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

let $n - 3k = 1$

$\frac{n-1}{3} = k$

$$T(n) = T(1) + \left(n - 3\left(\frac{n-1}{3} - 1\right)\right)^2 + \left(n - 3\left(\frac{n-1}{3} - 2\right)\right)^2 + \dots\ n^2$$

$$T(n) = T(1) + \left(n - [(n-1) - 3]^2\right) + (n - [n-1-6]^2 + (n - [n-1-9])^2 + \dots \dots + n^2$$

$$T(n) = 1 + [3+1]^2 + [6+1]^2 + \dots + n^2$$

$$T(n) = 1 + 4^2 + 6^2 + \dots + n^2$$

$$T(n) = n^2 + \dots + 1$$

$$T(n) = O(n^2)$$