DESIGN AND ANALYSIS OF ALGORITHM

Name - Priyanshu Ahlawat

Class - CST SPL 1

UNIVERSITY ROLL NO. - 20175 27

CLASS ROLL NO. - 12

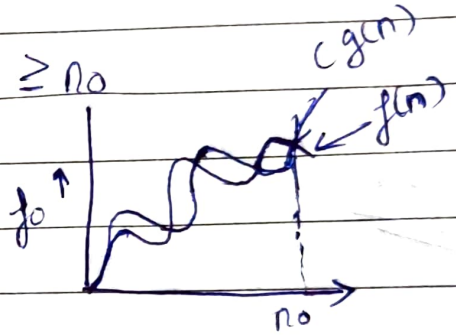
Ans 1 Asymptotic notation are the mathematical notation used to describe the running time of an algorithm when the input tend towards a particular value or a limiting value.

Types-

1 Big O - Gives worst case complexity.

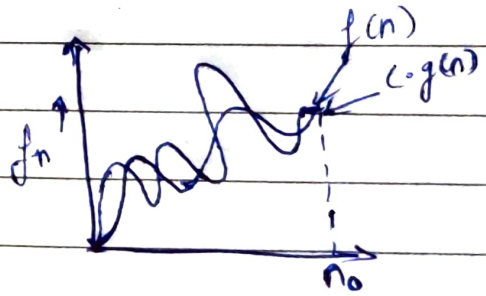
$f(n) = O(g(n))$
iff $f(n) \leq g(n)$ $\forall\ n \geq n_0$

for some constant $c > 0$
$\Rightarrow g(n)$ is tight upper bound of $f(n)$



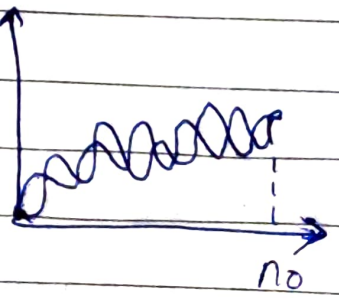
2 Big Omega (Ω) - Gives best-case complexity

$f(n) = \Omega(g(n))$
$g(n)$ is tight lower bound
$f(n) = \Omega(g(n))$
iff $f(n) \geq g(n)$
$\forall n \geq n_0$ for some constant $c > 0$



3 Theta (θ) - Gives average case complexity

$f(n) = \theta(g(n))$
$g(n)$ is both tight upper and lower bound of $f(n)$.
$f(n) = \theta(g(n))$



Ans 2

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

for loop will run for following values of i:

$$i = 2^0, 2^1, 2^2, \ldots\ldots 2^k$$

Let the final value will be n.

$$2^k = n$$

Taking $\log_2$ both sides

$$\log_2(2^k) = \log_2 n$$

$$k \log_2 2 = \log n$$

$$\boxed{k = \log n}$$

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

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

$T(n) = 3T(n-1) \quad \forall \quad n>0$

If we run a loop and i indicates the number of loops then

$\Rightarrow \quad i >= 0$

$\therefore \quad n > i$

$\Rightarrow \quad n-i > 0$

We can substitute $n-i$ for n

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

Considering few iterations -

for $i=0 \Rightarrow T(n) = 3T(n-1)$

$i=1 \Rightarrow T(n-1) = 3T(n-2)$

$i=2 \Rightarrow T(n-2) = 3T(n-3)$

Combining results ->

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

$\boxed{T(n) = O(3^n)}$, constant i.e 1 will be neglected

Ans

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

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

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

Substituting (2) in (1)

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

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

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

Similarly,

$T(n) = 4(2T(n-3)-1)-3 = 2^3 T(n-3) - 7$

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

let $n-k = 1$, because $n-k$ will become 1

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

$\Rightarrow \quad T(n) = O(2^k) \quad \because k = n-1$

$\Rightarrow \boxed{T(n) = O(2^n)}$ Ans

Ans 5

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

=> i is incrementing by one step
s is incrementing by value of i

Following will be values after few iterations -

=> i = 2, s = 3 1st iteration
=> i = 3, s = 6 2nd iteration
=> i = 4, s = 10 3rd iteration

Let the value of n be k.

Values of s = 1, 3, 6, 10 ---

s represents a series of sum of first n natural numbers

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

for stopping loop,

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

$$\boxed{T(n) = O(\sqrt{n})}$$ Ans

Ans 6

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

$i = 1, 2, 3 \ldots n$

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

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

$a_k = a + (k-1)d$

$a = 1, \quad d = 1$

$a_k <= \sqrt{n}$

$\sqrt{n} = 1 + (k-1) \cdot 1$

$\sqrt{n} = k$

$\boxed{T(n) = O(\sqrt{n})}$ Ans

Ans 7

```
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++;
}
```

$i = \frac{n}{2}$, $j = \log_2 n$, $k = \log_2 n$

$\left(\frac{n}{2} + 1\right)$ times $\log_2 n$ $\log_2 n$

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

$$= O\left(\left(\frac{n}{2}+1\right) \times (\log n)^2\right)$$

$$\boxed{T(n) = O(n(\log n)^2)}$$ Ans

8

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

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

$T(n) = T(n-3) + n^2$ — (1)

$T(1) = 1$ — (2)

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

$T(n-3) = T(n-6) + (n-3)^2$ — (3)

Put (3) in (1)

$T(n) = T(n-6) + (n-3)^2 + n^2$ — (4)

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

$T(n-6) = T(n-9) + (n-6)^2$ — (5)

put (5) in (4)

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

Generalising

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

Let $n-3k = 1$

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

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

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

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

$$T(n) = 1^2 + 4^2 + 7^2 \cdots n^2$$

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

$$\boxed{T(n) = O(n^2)}$$ Ans

Ans 9

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

for $i=1$, $j \rightarrow n$ times

for $i=2$, $j = 1+3+5 \cdots + n$

$a_n = a + (k-1)d$

$a = 1, \quad d = 2$

$\therefore n = 1 + (k-1) \times 2$

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

$$k = \frac{n-1}{2} + 1$$

$$\boxed{k = \frac{n+1}{2}}$$ No. of terms

for $i = 2$, $j \to \frac{n+1}{2}$ times

for $i = 3$, $j = 1 + 4 + 7 + \dots n$

$n = 1 + (k-1) \times 3$

$$\boxed{\frac{n-1}{3} + 1 = k}$$ No. of terms

for $i = 3$, $j = \frac{n+2}{3}$ times

Generalising

for $i = n$, $j = \frac{n+k-1}{k}$ times

Time complexity is

$$\underbrace{n + \frac{n+1}{2} + \frac{n+2}{3} + \dots + \frac{n+k-1}{k}}_{n \text{ terms}}$$

General term = $\frac{n+k-1}{k}$

$$\sum_{k=1}^{n} \frac{n+k-1}{k} \Rightarrow \frac{\sum_{1}^{n} n + \sum_{1}^{n} k - \sum 1}{k}$$

$$\Rightarrow \frac{\frac{n(n+1)}{2} + nk - n}{k}$$

$$\Rightarrow \frac{n^2 + \frac{n}{2} + nk - n}{k}$$

$$T(n) = \frac{n^2 + \frac{n}{2} + nk - n}{k}$$

Neglecting constant terms

$$\boxed{T(n) = O(n^2)}$$ Ans

Ans 10 Given -> n^k & c^n

Relation between n^k and c^n is

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

as $n^k \leq a c^n$

$\forall$ $n \geq n_0$ & some constant $a > 0$

for $n_0 = 1$
$c = 2$

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

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