DAA assignment - 1

Name: Karan Maurya
C. Roll: 11
U. Roll: 2016806

Ans. 1. **Asymptotic Notation:**

These notation are used to tell the complexity of an algorithm with respect to the input size.

Different asymptotic notation:

1. **Big oh (O):**

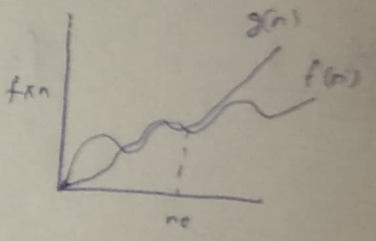
It is a upper tight bound of $f(n)$.

$f(n) \leq c \cdot g(n)$

$f(n) = O(g(n))$

Ex: $g(n) = n^2 + n + 5$

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



2. **Big omega (Ω):**

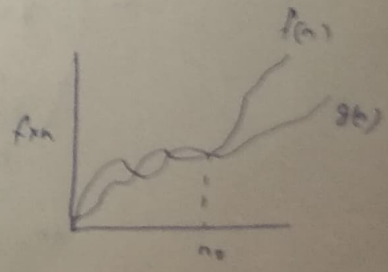
It is a lower tight bound of $f(n)$.

$f(n) \geq c \cdot g(n)$

$f(n) = \Omega(g(n))$

Ex: $g(n) = n^2 + \log n$

$\boxed{f(n) = \Omega(\log n)}$

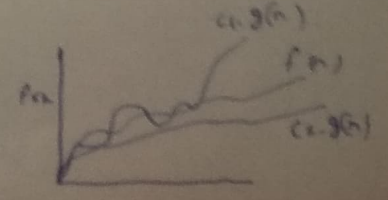


3. **Theta (θ):**

It give us a range of $f(n)$.

$c1 \cdot g(n) \leq f(n) \leq c2 \cdot g(n)$

$\boxed{f(n) = \theta(g(n))}$

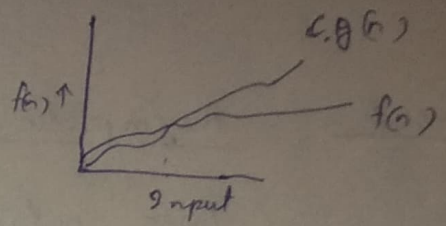


4. small oh (o):

It is a upper bound of f(n).

$f(n) = o(g(n))$

$f(n) < c.g(n)$

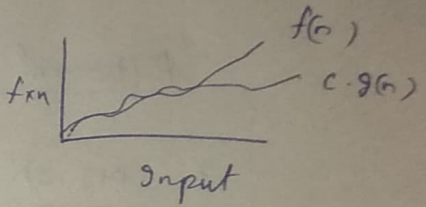


5. Small omega (ω)

It is a lower bound of f(n).

$f(n) = \omega(g(n))$

$f(n) > c \cdot g(n)$



Ans. 2.

```
for(i=1; i<=n; i=i+1)
    i = i*2;
```

i
1
2+1 = 3
3*2+1 = 7
7*2+1 = 15
⋮
n

K

So, $2^K - 1 = n$

$2^K = n+1$

$\log(n+1) = K$

So, $\boxed{f(n) = O(\log n)}$

So, it has logarithmic complexity.

Ans. 3 $T(n) = \begin{cases} 3T(n-1) & \text{if } n > 0 \\ 1 & \text{if } n \leq 0 \end{cases}$

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

$T(0) = 1$

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

$T(n) = 9T(n-2)$

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

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

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

at $T(0) = 1$

$n-1-k = 0$

$k = n-1$

So, $T(n) = 3^{n-1}$

$$T(n) = \Theta(3^n)$$

Ans 4. $T(n) = \begin{cases} 2T(n-1) - 1 & \text{if } n > 0 \\ 1 & \text{if } n \leq 0 \end{cases}$

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

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

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

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

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

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

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

$$T(n) = 2^k (T(n-k) - 2^{k-1} - 2^{k-2} - 2^{k-3} - \cdots - 2 - 1$$

$$T(n) = 2^k - (1 + 2 + 4 + \cdots + 2^{k-1})$$

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

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

$$\boxed{T(n) = 1}$$

Ans. 5

```
int i=1, s=1;          — (2)
while (s<=n){          ...
    i++;               — (n)
    s = s+i;           — (n)
    printf("#");       — (n)
}
```

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

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

Ans. 6

```
void function (int n)          — (1)
{
    int i, count=0;            — (2)
    for(i=1; i*i<=n; i++)      ⌐ O(√n)
        count++
}
```

$$k \begin{cases} i \\ 1 \\ 2+2 = 4 \\ 5+5 = 25 \\ \vdots \\ n \end{cases}$$

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

$$\boxed{k = \sqrt{n} - 1}$$

$$\boxed{\text{Time complexity } [T(n) = O(\sqrt{n})]}$$

Ans. 7.

```
void function (int n)                    - (1)
{
    int i, j, k, count = 0;              - (4)
    for ( i = n/2 ; i <= n ; i++)        - (n/2)
        for(j=1 ; j<=n ; j = j*2)        - (log n)
            for (k=1; k<=n; k = k*2)     - (log n)
                count++;
}
```

$$T(n) = \frac{n}{2} \log n \log n + 5$$

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

Ans. 8.

```
                                  ← T(n)
function (int n) {
  if (n == 1)
    return;                       - (1)
  for(i=1 to n) {
     for (j=1 to n) {
        printf("*");              - (n^2)
     }
  }
  function (n-3);                 ← T(n-3)
}
```

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

$$T(0) = 1$$

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

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

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

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

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

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

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

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

$$T(n) = 1 + 3^2 + 6^2 + 9^2 + 12^2 + \cdots + (3(k+1))^2$$

$$T(n) = 1 + \sum_{i=0}^{k} [3(i+1)]^2$$

$$T(n) = 1 + \sum_{i=0}^{\frac{n-3}{3}} [3(i+1)]^2$$

$$\boxed{T(n) = \Theta(n^3)}$$

Ans. 9:

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

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

Ans. 10

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

$$n^k \leq c_1 \cdot c^n$$

Let, $n_0^k \leq c_1 \cdot c^{n_0}$