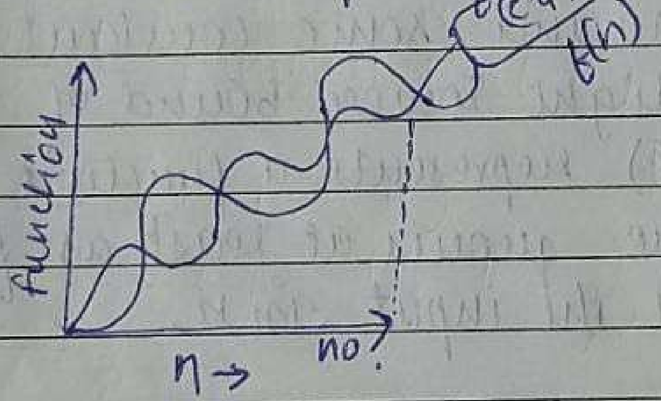
MD Rishitosh Khan
4th Sem
Section - CE - SPL
class Roll no. 09

Date ___/___/___

DAA Assignment - 1

1) Asymptotic notation is used to describe the behaviour of a function as its input grows without bound. It is commonly used in the analysis of algorithms to describe their time and space complexity input is very large
Asymptotic means $\rightarrow$ tending to infinity.

1) Big O (O) : It spends the upper bound of a function. It is used to describe the worst case scenario for an algorithm.
It represents the maximum time an algorithm will take to complete for any given input size.

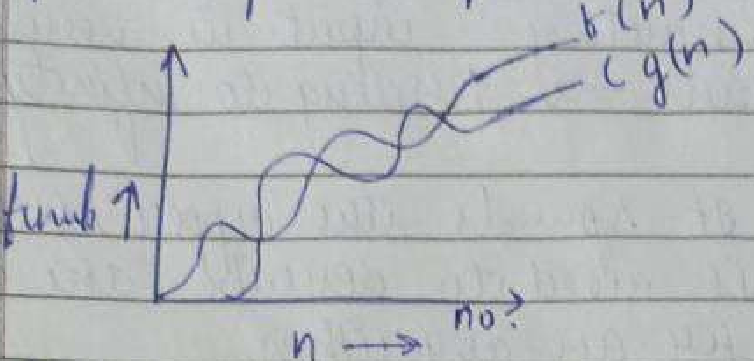


$$f(n) = O(g(n))$$
iff. $f(n) \leq c \cdot g(n)$

Example: $O(n)$ represents a function whose running time grows linearly with the input size n.

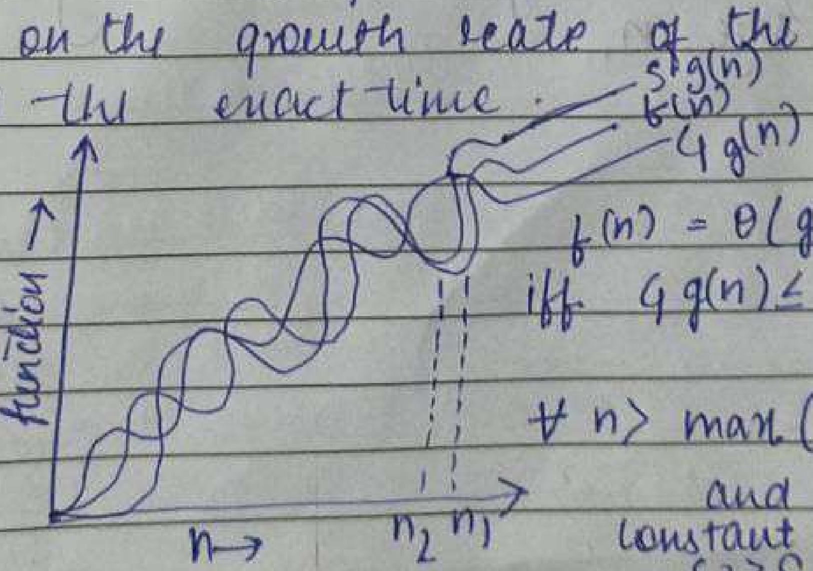
Date

2 Omega notation (Ω): It represents the lower bound of a function.
It is used to describe the best case scenario for an algorithm.
It provides a lower bound on the growth of the function:



$f(n) = \Omega(c\,g(n))$
iff $f(n) \geq c\,g(n)$
$\forall$ $n \geq n_0$ and some constant $c > 0$
* g(n) is tight lower bound of f(n)
example: $\Omega(n)$ represents a function whose running time grows at least as fast as linearly with the input size n.

3) Theta notation (Θ): It represents both the upper and lower bounds of a function.
It is used to describe the tight bound of an algorithm. It provides an exact bound on the growth rate of the function
It tells the exact time.



$f(n) = \Theta(g(n))$
iff $c_1 g(n) \leq f n \leq c_2 g(n)$
$\forall$ $n > \max(n_2, n_1)$
and some constant $c_1 > 0$ &
$c_2 > 0$

Date ___/___/___

example :- $O(n)$ represents a function whose running time grows linearly with the input size.

2 => for (i 1 to n)
{ $i = i * 2$

}

$1, 2, 4, 8, 16, 32$ - - - - - - K terms

K_{th} term $= a \cdot r^{K-1}$ $\left\{ r = \frac{4}{2} = 2 \right\}$ $a = 1$; ,

K_{th} term $\Rightarrow 1 \cdot 2^{K-1}$

$n = 2^{K-1}$

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

Taking log 2 both sides

$\log_2 2n = \log_2 2^K$

$\log_2 2n = K$ $\{\log_2 2 = 1\}$

$K = \log_2 2 + \log_2 n$

$K = \log_2 2 + \log_2 n$

$K = 1 + \log_2 n$

complexity $\Rightarrow O(1 + \log_2) n)$, ignoring constants we have

Ans $= O(\log_2 n)$.

3 =>

$T(n) = 3T(n-1)$ —— ①

$T(10) = 1$ —— ②

Putting, $n = n-1$ in ①

$T(n-1) = 3T(n-2)$ —— ③

Putting, $n = n-2$ in ①

$T(n-2) = 3T(n-3)$ —— ④

Putting, $n = n-3$ in ①

$T(n-3) = 3T(n-4)$ — ⑤

Putting ⑤ and ④

$T(n-2) = 9T(n-4)$ — ⑥

Putting ⑥ in ③

$T(n-1) = 3^3 T(n-4)$ — ⑦

Putting ⑦ in ①

$T(n) = 3^4 T(n-4)$ — ⑧

for K terms,

$T(n) = 3^K T(n-K)$ — ⑨

Assume $n-K = 0$

$n = K$

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

$T(n) = 3^n$

complexity $\Rightarrow$ $\underline{\underline{O(3^n)}}$

$\underline{\underline{4}} \Rightarrow$ $T(n) = 2T(n-1)-1$ — ①

$T(0) = 1$

Putting $n = n-1$ in ①

$T(n-1) = 2T(n-2) - 1$ — ③

Putting $n = n-2$ in ① — ~~④~~

$T(n-2) = 2T(n-3) - 1$ — ④

Putting $n = n-3$ in ①

$T(n-3) = 2T(n-4) - 1$ — ⑤

Putting ⑤ in ④

$T(n-2) = 2 \cdot 2T(n-4) - 2 - 1$ — ⑥

Putting ⑥ in ③

$T(n-1) = 2 \cdot 2 \cdot 2\, T(n-4) - 2 - 2 - 2 - 1$ — ⑦

Putting ⑦ in ①

$T(n) = 2^4 T(n-4) - 2^3 - 2^2 - 2^1 - 2^0$ — ⑧

for K terms.

Date ___/___/___

$T(n) = 2^k T(n-k) - 2^{k-1} - 2^{k-2} - 2k^{2} \cdot \cdot - 2^1 - 2^0$ — $\boxed{9}$

Assume $n-k=0$

$\Rightarrow 2^k - T(0) - 2^{k-1} - 2^{k-2} - 2^{k-3} - - - 2^0$

$\Rightarrow 2^k - 2^{k-1} - 2^{k-2} - 2^{k-3} - - - - \cdot - 2^1 - 2^0$

$\Rightarrow 2^k - (2^0 + 2^1 + - - - - - \cdot 2^{k-3} + 2^{k-2} + 2^{k-1})$

$\Rightarrow 2^k - \left(\frac{1-2^k}{1-2}\right) \Rightarrow 2^k + 1 - 2^k.$

$\Rightarrow 1$

complexity = $\underline{\underline{O(1)}}$

$\underline{\underline{5}} \Rightarrow$

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

print("#") will execute till $s <= n$ and s is incremented by i every time which also increases by 1 everytime

$i = 1, 2, 3, 4, 5, 6, 7 - - - - - -$

$S = 1, 3, 6, 10, 15, 21, 28 - - \quad - - - \cdot k^{th}$ term.

loop will run till $s <= n$

S is the sum of K numbers (AP)

k^{th} term $= \frac{k * (k+1)}{2}$

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

$2n = k^2 + k$

$k(1+k) = 2n$

$k^2 + k - 2n = 0.$

Date ____

$k = \left(-\frac{1}{2} \pm \sqrt{1+4*n}\right)/2$

$k = 2\sqrt{n}$

$k = \sqrt{n}$

$\Rightarrow O(\sqrt{n})$

6 →

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

$\Rightarrow 1, 2^2, 3^2, 4^2, 5^2$ - - - - - - kth

kth term = $k * k$

kth term $<= n$

$k * k <= n$

$k^2 = n$

$k = \sqrt{n}$

complexity $\Rightarrow O(\sqrt{n})$.

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 *
            {
```

Date ___/

$n/2, \frac{n}{2}+1, \frac{n}{2}+2, \frac{n}{2}+3, \ldots\ldots$ kth ..

kth term $= \frac{n}{2}+k$

$n = \frac{n}{2}+k$. $\Rightarrow k = n - \frac{n}{2}$

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

this loop will run $n/2$ times

Total complexity $\Rightarrow \frac{n}{2} * (1+\log_2 n) * (1+\log_2 n)$

$\Rightarrow \frac{n}{2} + \frac{n}{2}(\log_2 n + \frac{n}{2}(\log_2 n)^2 + \frac{2}{2}\log_2 n$

$\Rightarrow O(n(\log_2 n)^2)$

i	j	k
$n/2$	$1 \to n$	$1+\log_2 n$
$n/2+1$	$1 \to n$	$1+\log_2 n$.
$n/2+2$	$1 \to n$	
n	$1 \to n$	

8 →

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

i	j
1	$1 \to n$
2	$1 \to n$
3	$1 \to n$
4	$1 \to n$
n	n tim.
n tim	

$\Rightarrow n * n$-times.

Date ___/___/___

count++ ;
} } } }

Time complexity of inner most loop;
k = 1 to n, k = k * 2.
1, 2, 4, 8, 16 - - - - - - . kth Term
kth term = 2^{k-1}
$n = \frac{2^k}{2}$ => $2n = 2^k$ =>

Taking $\log_2$ both sides
$\log_2 2n = \log_2 2^k$
$\log_2 2n = k$
$k = \log_2 2 + \log_2 n$

$\boxed{k = 1 + \log_2 n}$
It means for each value of j, this loop runs $1 + \log_2 n$ times.

complexity of middle loop,
j = 1 to n; j = j×2
1, 2, 4, 8, 16 - - - - . kth term.
$(1 + \log_2 n)$.

It means for each value of i this loop runs $(1 + \log_2 n)$ times
complexity of outer most loop;
i, n/2 to n, i++

Date ___/___/___

for function(n-3)

n, n-3, n-6, n-9 - - - - - kth term

n, n-1·3, n-2·3, n-3·3 - - - - kth term

kth term = $n-(k\cdot 1)\cdot 3 = n-3k-3$

$1 = n-3k-3$

$\Rightarrow n-3k-3-1=0 \Rightarrow n-3k-4=0 \Rightarrow k = \frac{n-4}{3}$

Inner most loop will execute = $n*n*\frac{n-4}{3}$ times

$= \frac{n^3-4n^2}{3}$

compl = $O(n^3)$ ignore constants and smaller values.

9 →

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

Outer loop will ~~r~~ n times (i).

for $i=1$, j will run n times

$i=2$, j will run n/2 times

$i=3$, j will run n/3 times

$i=n$, j will run n/n times

Inner loop will run $= \left(n + \frac{n}{2} + \frac{n}{3} + \cdots \frac{n}{n-1} + \frac{n}{n}\right)$ tim

$\Rightarrow n\left(1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} \cdots + \frac{1}{n}\right)$

$\Rightarrow n \cdot \log n$

sum of $\left(1 + \frac{1}{0} + \frac{1}{3} + \frac{1}{4} \cdots + \frac{1}{n}\right)$ is $\log n$.

Comple $\Rightarrow O(n \cdot \log n)$