Lecture-2 Sunita Gorhwal

Time-Complexity

Complexity :- Complexity of an algorithm M is the function $f(n)$ which gives the running time and / or storage space requirement of the algorithm in term of the size n of the input data.

$\Rightarrow$ the storage space required by an algorithm is simply a multiple of the data size n.

$\therefore$ the term "Complexity" shall refer to the running time of the algorithm.

① Consider the following logic for calculating sum of natural no.'s.

```
for( i=0 ; i<=n ; i++)
{
    sum = sum+i ;
}
```

Observations are :-

		How many time the control goes.	Assuming time taken per unit.	time taken each by instruction.
• Assignment	i=1	1	1 unit	1 * 1 = 1
• Comparison	i<=n	n+1	1 unit	(n+1) * 1 $\Rightarrow$ n+1
• increment	i++	n	1 unit	n * 1 $\Rightarrow$ n
• Addition	sum+i	n	1 unit	n * 1 = n
• Assignment	sum=sum+i	n	1 unit	n * 1 = n
				1+n+1+n+n+n

Total time $\Rightarrow 4n+3$

Scanned by CamScanner

$f(n) = 6n + 3$

According to Big-O Notation
We have to ignore the implementation dependent factors & eliminate the constant factors.

$f(n) = 6n+3 \approx 6n \approx O(n)$.

The cases which ~~one~~ one usually investigates in complexity theory are :-

- Worst Case :→ the maximum value of $f(n)$ for any possible input
- Average Case :- the expected value of $f(n)$
- Best case :- the minimum value of $f(n)$

⇒ the complexity of an algorithm shall mean the function which gives the running time of the worst case in terms of the input size.

We use Big-O Notation, to compare the function $f(n)$ with the standard function

Big-O ⇒ means on the order of

ie $O(n)$ ⇒ means on the order of n

Expressing time Complexity By $f(n)$

$f(n)$	$O(g(n))$
2 unit	$O(1)$
$\frac{1}{2}n^2 + 3$	$O(n^2)$
$500n^2 - 25$	$O(n^2)$
$n^3 + n^2 - 1$	$O(n^3)$
$5n^3 + 3n + 6$	$O(n^3)$

Problems on Complexity :-

(1)
1 unit ← c = a ;
1 unit ← a = b ;
1 unit b = c ;

} Assignment statement so they will take same constant time to execute.

total time ⇒ 3 unit ⇒ Constant value. In Big-O Constant term is denoted by $O(1)$.

(2)
if (a > b) → we are checking this instruction only once
printf ("a is larger");
else
printf ("b is larger");

⇒ These are I/O instructions so we don't consider the time taken by them.

Scanned by CamScanner

as if (a>b) ⇒ control only goes once there so it will take some constant unit of time.
say it is 2 unit

≈ 2 ≈ O(1)

(3) . Logic of linear Search

```
for( i=0 ; i<n ; i++)
{
  if (a[i] == ele)
  {
    Prinf("element found");
    break;
  }
}
```

Observations

		How many time the control goes.	Assuming one time taken per unit	time taken by each instruction
• Assignment	i=0	1 (once)	1 unit	1×1 = 1
• Comparison	i<n	n times	2 unit	n*2 = 2n
• increment	i++	n times	2 unit	n*2 = 2n
• if (condition)		n times	1 unit	n*1 = n
• Body of if.		once when element found. (1)	1 unit	1*1 = 1
			Total time ≈	1+2n+2n+n+1

≈ 5n+2

Scanned by CamScanner

So $f(n) = 5n+2$

According to Big-O Notation

$f(n) = 5n+2 \approx 5n \approx O(n)$.

Q4

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

Assuming $n = 1024$

So for which value of i we will go inside the loop

$i = 1, 2, 4, 8, 16, 32, 64, 128, 256, 512$.

ie we will have 10 iterations.

$2^{10} = 1024$

$\log_2 1024 = 10$

∴ $\log_a x = m$ then $a^m = x$

taking 1024 as n ie $\log_2 n = 10$

$\approx O(\log_2 n)$.

Scanned by CamScanner

Q5 `for(i=n ; i>=1 ; i=i/2)`

{
=
}

Assuming $n = 1024$

∴ for which values of i, we go inside the loop

$i = 1024, 512, 256, 128, 64, 32, 16, 8, 4, 2, 1$

ie total 11 iterations

∴ $\log_2 1024 = 10$

$\log_2 n \Rightarrow 10$ (11 iterations or 9 iterations will not make any drastic change while computing the time complexity)

$\approx O(\log_2 n)$

Q6 Independent loops

`for ( i=1; i<=n ; i++)` → this will execute $\approx n$ times.

{
=
}

`for (j=1 ; j<n ; j=j*2)` → this will execute $\approx \log_2 n$ times.

{ =
}

Scanned by CamScanner

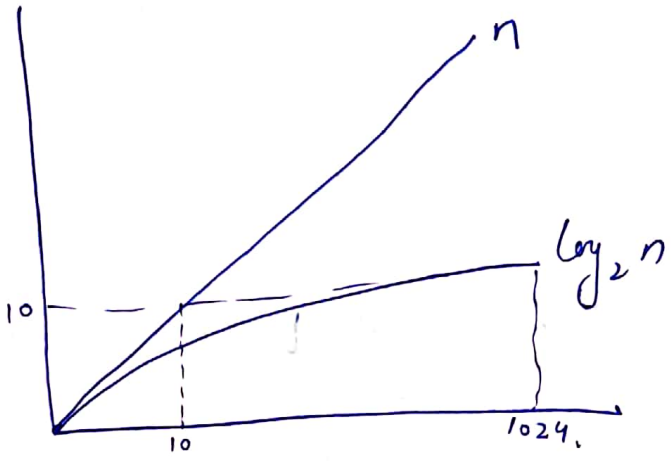
for the whole function., we have to add these

$\Rightarrow n + \log_2 n$.

Acc. to Big-O, small values will be neglected)

$\Rightarrow n + \log_2 n$

$\Rightarrow O(n)$



As you can observe through graph that n is larger than $\log_2 n$ ∴ in case of $\log_2 n$ we will get 10 when it will reach 1024.
But in case of O(n) we get 10 on 10.

Dependent loop

Q7
```
for (i=1; i<=n; i++)
{
  ≡
  for(j=1; j<=n; j++)
  {
  ≡
  }
}
```

Observation:

i=1 then j=1, 2, ---- n times
i=2 then j=1, 2, ---- n time
⋮
i=n then j=1 to ------ n times

Scanned by CamScanner

so j loop will execute $n \times n \Rightarrow n^2$ times.

& i loop will execut n times.

∴ total time taken is

$$n + n^2$$

Acc. to Big-O smaller values are neglected.

∴ $n + n^2 \simeq O(n^2)$

Q8

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

Body of outer loop →

$i=1$, $j = 1, 2, 4, \ldots$ ie $\log_2 n$.

$i=2$ j $j = 1 \ldots$ ie $\log_2 n$.

Therefore

$i=n$ j then $j = 1 \ldots \log_2 n$.

So j in loop will execute $n * \log_2 n$ times

i in loop will execute n times

total time taken

$n + n * \log_2 n \simeq O(n \log_2 n)$

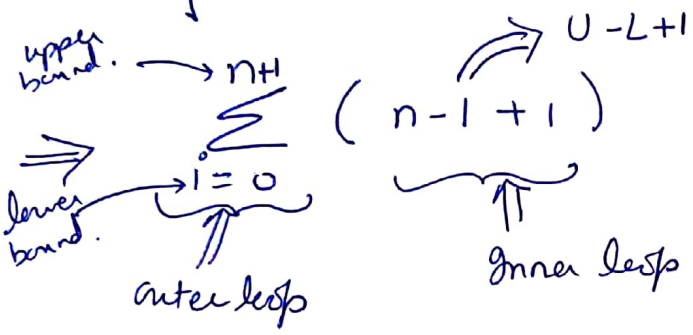
Scanned by CamScanner

For Dependent loops

Q1

```
for(i=0 ; i<=n ; i++)
{
  for (j=1 ; j<=n ; j++)   // for the inner most loop we can use U - L + 1
  {
  }
}
```

U = upper bound, L = lower bound.



$$\Rightarrow \sum_{i=0}^{n} (n-1+1) \Rightarrow \sum_{i=0}^{n} n$$

n+1 times.

After opening the $\sum$ $\Rightarrow$ $n + n + n - - - -$ (i.e. for $i=0$ $i=1$ $i=2$ ... $i=n$.)

$$\Rightarrow n(n+1)$$

$$\Rightarrow n^2 + n$$

According to Big-O neglect the smaller one.

$$\approx O(n^2)$$

Scanned by CamScanner

Q2.
```
for (i=1; i<=n; i++)
{
    for (j=1; j<=i; j++)
    {
        ≡
    }
}
```

→ I am taking here n not $n+1$ as +1 or -1 will not affect the overall complexity.

$$\underbrace{\sum_{i=n}^{n}}_{\text{outer loop}} \underbrace{(i-1+1)}_{\text{inner loop.}}$$

$$\sum_{i=1}^{n} i \Rightarrow 1+2+3+\cdots+n \text{ times.}$$

natural no. of sum.

$$\frac{n(n+1)}{2} \approx \frac{n^2}{2} + \frac{n}{2}$$

$$\approx \frac{n^2}{2} \approx O(n^2)$$

Q3.
```
for (i=1; i<=n; i++)
{
    for (j=1; j<=n; j++)
    {
        for (k=1; k<=n; k++)
        {
            ≡
        }
    }
}
```

always apply U-L+1 for the innermost

$$\underbrace{\sum_{i=1}^{n}}_{\text{outermost}} \underbrace{\sum_{j=1}^{n}}_{\text{middleone}} \underbrace{(n-1+1)}_{\text{innermost}}$$

Scanned by CamScanner

$$\sum_{i=1}^{n}\left(\sum_{j=1}^{n}(n)\right)$$

$$\sum_{i=1}^{n}(n+n+\text{----- n times})$$

$$\sum_{i=1}^{n} n(n) \Rightarrow \sum_{i=1}^{n} n^2$$

$\Rightarrow$ $n^2 + n^2 + n^2 + \text{---- n times}$

for $i=1$ $i=2$ $i=3$ ----

$\Rightarrow$ $n^2 * n \Rightarrow n^3 \approx O(n^3)$

Q4 $\sum_{i=1}^{n}\sum_{j=1}^{n}(n-1)$

$$\sum_{i=1}^{n}\left(\sum_{j=1}^{n} n - \sum_{j=1}^{n} 1\right)$$

$$\sum_{i=1}^{n}(n+n+\text{---n times} - 1+1+\text{---- n times})$$

$$\sum_{i=1}^{n}(n \cdot n - 1 \cdot n)$$

$$\sum_{i=1}^{n}(n^2 - n) \Rightarrow \sum_{i=1}^{n} n^2 - \sum_{i=1}^{n} n$$

$\Rightarrow$ $(n^2 + n^2 + \text{---- n times}) - (n+n+\text{---- n times})$

$\Rightarrow$ $(n^2 \cdot n) - (n \cdot n)$

$\Rightarrow$ $n^3 - n^2 \approx O(n^3)$ ($\because$ smaller will be neglected)

Scanned by CamScanner

Q5

```
for (i=0; i<=n; i++)
for (j=0; j<=i*i; j++)
for (k=0; k<=j; k++)
```

$$\sum_{i=0}^{n} \sum_{j=0}^{i^2} (j-0+1)$$

$$\sum_{i=0}^{n} \sum_{j=0}^{i^2} (j+1)$$

$$\sum_{i=0}^{n} (0+1+2 \text{ --- } i^2 \text{ times}) + \sum_{i=0}^{n} (1+1+ \text{ --- } i^2 \text{ times})$$

↑ natural no. sum in form of i^2

$$\sum_{i=0}^{n} \left(\frac{i^2(i^2+1)}{2}\right) + \sum_{i=0}^{n} i^2 * 1$$

$$\sum_{i=0}^{n} \frac{i^4}{2} + \sum_{i=0}^{n} \frac{i^2}{2} + \sum_{i=0}^{n} i^2$$

$$\frac{1}{2}\left[0^4+1^4+ \text{ ----- } n^{4\text{th time}}\right] + \frac{1}{2}\left[0^2+1^2+ \text{ ---- } n^{2\text{nd time}}\right] + \left[0^2+1^2+ \text{ -- } n^{\text{th}}\right]$$

$$\frac{1}{2}\left[\frac{n(n+1)(2n+1)(3n^2+3n-1)}{30}\right] + \underbrace{\frac{1}{2}\left[\frac{n(n+1)(2n+1)}{6}\right] + \left[\frac{n(n+1)(2n+1)}{6}\right]}$$

↑ They will be smaller so we eliminated. mem.

Scanned by CamScanner

$\Rightarrow \frac{(n^2+n)(2n+1)(3n^2+3n-1)}{30}$

$\Rightarrow \frac{(2n^3+n^2+2n^2+n)(3n^2+3n-1)}{30}$

$\Rightarrow 6n^5+6n^4-2n^3+9n^4+9n^3-3n^2+3n^3+3n^2-n$

$\Rightarrow 6n^5+15n^4+10n^3-n$ [smaller will be neglected]

$\Rightarrow 6n^5$ [Constant will be ignored]

$\Rightarrow O(n^5)$

Q6
```
for(i=1 ; i<=n ; i++)
for (j=1 ; j<=i ; j++)
for (k=1 ; k<=j ; k++)
```

$\sum_{i=1}^{n} \sum_{j=1}^{i} (j-1+1) \Rightarrow \sum_{i=1}^{n} \sum_{j=1}^{i} (j)$

$\Rightarrow \sum_{i=1}^{n} (1+2+3+\cdots \text{ i time})$

$\sum_{i=1}^{n} \frac{i(i+1)}{2} \Rightarrow \sum_{i=1}^{n} \left(\frac{i^2}{2}+\frac{i}{2}\right)$

$\Rightarrow \sum_{i=1}^{n} \frac{i^2}{2} + \sum_{i=1}^{n} \frac{i}{2}$ — smaller so neglected.

$\Rightarrow 1^2+2^2+3^2+\cdots$ n times.

$\Rightarrow \frac{n(n+1)(2n+1)}{6} \approx n^3+n^2+n \approx O(n^3)$

Scanned by CamScanner

Q7

```
for( i=0 ; i<n ; i++)
{
    for( j= n-1 ; j>=i ; j-- )
    {
        =
    }
}
```

$$\sum_{i=0}^{n-1} (n-1-i+1) \Rightarrow \sum_{i=0}^{n-1} n - \sum_{i=0}^{n-1} i$$

(upper, lower)

$\Rightarrow$ n + n + ---- n-1 times $- \sum_{i=0}^{n-1} i$

~~natural no. sum in form of (n-1) time.~~ ~~smaller so neglected.~~

$\Rightarrow (n * n-1) - (0+1+2+ \cdots n-1 \text{ time})$

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

$\Rightarrow n^2 - n - \frac{n^2}{2} + \frac{n}{2}$ [smaller will be neglected]

$\Rightarrow n^2 - \frac{n^2}{2}$

$\Rightarrow O(n^2)$

Q8

```
for( i=0 ; i<n ; i++)
{
    for(j=0 ; j<= i*i ; j++)
    {
        =
    }
}
```

$$\sum_{i=0}^{n-1} (i^2 - 0 + 1) \Rightarrow \sum_{i=0}^{n-1} (i^2) - \sum_{i=0}^{n-1} 1$$

Scanned by CamScanner

$$\sum_{i=0}^{n-1} i^2 - \sum_{i=0}^{n-1} 1$$

$(0^2 + 1^2 + \cdots \text{ } n-1 \text{ time}) - (1 + 1 + \cdots n-1 \text{ time})$

$$\Rightarrow \frac{(n-1)(n-1+1)(2n-2+1)}{6} + 1(n-1)$$

$$\Rightarrow \approx O(n^3)$$

Q9

```
for (i=1; i<n; i++)
{
    for (j=i; j>=0; j=j/2)
    {
    }
}
```

$$\sum_{i=1}^{n-1} \log_2 i$$

$$\Rightarrow \log_2 1 + \log_2 2 + \cdots \log_2 (n-1)$$

$$\Rightarrow \log (1 \cdot 2 \cdot 3 \cdots n-1)$$

$$\Rightarrow \log (n-1\,!)$$

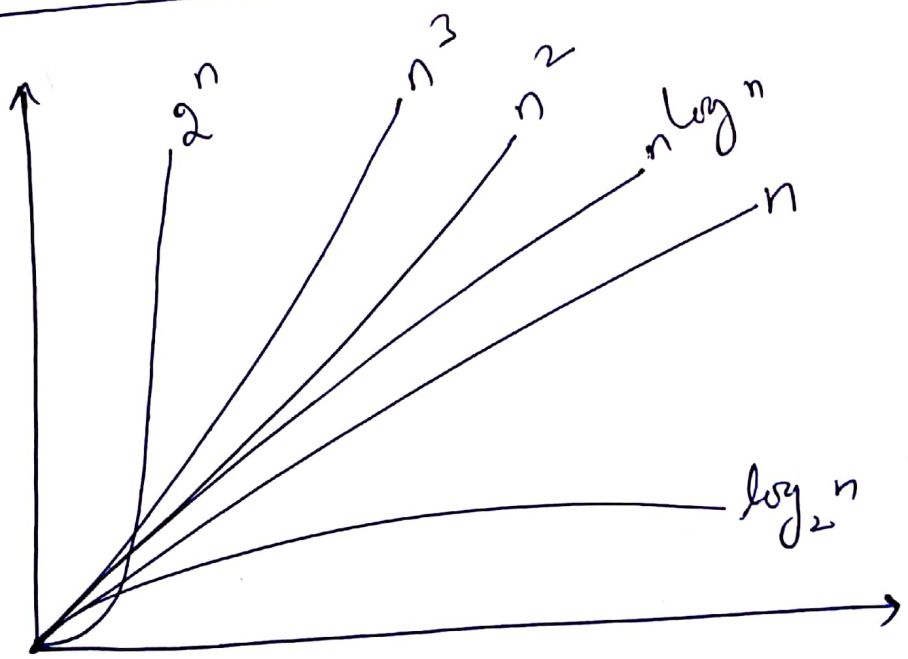
$$\Rightarrow \log((n-1)(n-2)(n-3)\cdots 1)$$

$$\Rightarrow \log_2 n^n \Rightarrow n \log_2 n$$

Scanned by CamScanner

Rate of Growth of standard functions

n / g(n)	$\log n$	n	$n \log n$	n^2	n^3	2^n
5	3	5	15	25	125	32
10	4	10	40	100	10^3	10^3
100	7	100	700	10^4	10^6	10^{30}
1000	10	10^3	10^4	10^6	10^9	10^{300}



Scanned by CamScanner