⇒ Time Complexity (Time consumed by a program to execute entirely)

↳ Relationship between Input and running time

O = capital O notation

worst case senario

ex:-

```
main() {
    Syso("Hello");   // 1
    Syso("Hi");      // 1
}
```

operations = 2

Time Complexity = O(2)

≅ O(1) constant

ex:-

```
main() {int n = scn.nextInt();
    for( int i=0; i<n; i++) {
        Syso("Hi");
    }
}
```

operation = n

Time Complexity ≅ O(n)

"n" is the input from user

⇒ Types of operations

- linear equation
- quadratic equation
- cubic equation
- Logerethmic equation
- constant equation

:

Input	output operation
n	n
n	n^2
n	n^3
n	$\log(n)$
n	1

Ex 1 :-

```
public static void main() {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    for(int i=0; i<n; i++) {
        Syso("Hello");
    }
}
```

n times

output operations

input :-

n = 1 ⟶ 1

n = 10 ⟶ 10

n = 90 ⟶ 90

n = 10000 ⟶ 10000

input $\propto$ n

$\hookrightarrow O(n)$

linear equation

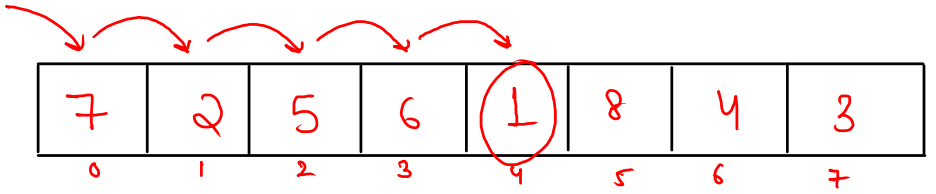
⟹ Complexity (Time) :-

1) Best case :- when least no. of operations has to be done

2) average case :- when average no. of operations has to be done

3) worst case :- when most no. of operations has to be done

Ques You have an array and you have to find '1' in it.




```
int n=8;
for(int i=0; i<n; i++){
    if(arr[i]==1){
        return;
    }
}
```

Time Comp. = O(n)
where 'n' is size of array

operation = 8 (worst case)

operation = 1 (best case)

operations = 5 (average case)

Ex 1 :-

```
public static void main() {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    for(int i=0; i<n; i++){
        for(int j=0; j<n; j++){
            Syso("Hi");
        }
    }
}
```

n = 3

i=0, j=0
j=1
j=2
} 3

i=1, j=0
j=1
j=2
} 3

i=2, j=0
j=1
j=2
} 3

Input	output ope.
n = 5	⟶ 25
n = 6	⟶ 36
n = 10	⟶ 100

$$\text{Input} \propto n^2$$

↳ $O(n^2)$

Quadratic equation

Ex 2 :-

```
public static void main() {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int m = scn.nextInt();
    for(int i=0; i<n; i++) {
        for(int j=0; j<m; j++) {
            Syso("Hi");
        }
    }
}
```

Input α $n * m$

$O(n*m)$

Operations = $n * m$

$n = 2, \; m = 3 \rightarrow o/p = 6$

Ex:-

```
main() {
    Scanner scn = new Scanner();
    int n= scn.nextInt();
    int m= "      "  ;

    for(int i=0; i<n; i++) {
        Syso("Hi1");          //3
    }

    for (int j=0; j<m; j++) {
        Syso("Hi2");          //4
    }
}
```

$n = 3$

$m = 4$

operation :-

3+4

Time Compl.:-

$\Rightarrow O(n+m)$

e.g., $n = 2$

$m = 2000000$

operation = 2000002

$\simeq O(m)$ $m >>> n$

$\simeq O(n)$ $n >>> m$

Note:- whenever we have multiple values in addition then we ignore all the smaller values and only the largest one will be considered

Time complexity :- $O(n+m+a+b)$

Case :-

$a = 7$

$b = 2$

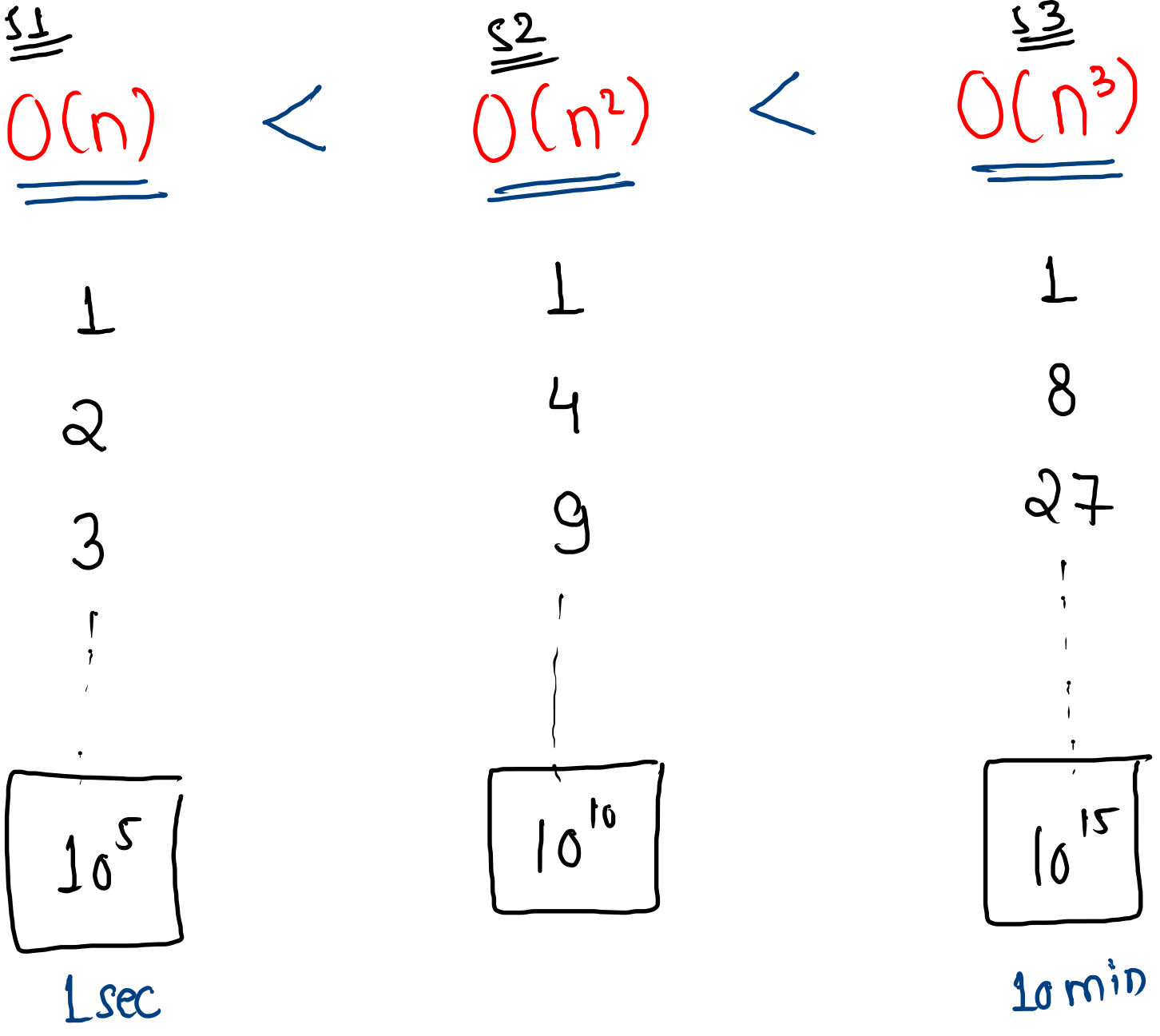
$n = 6$

$m = 10$

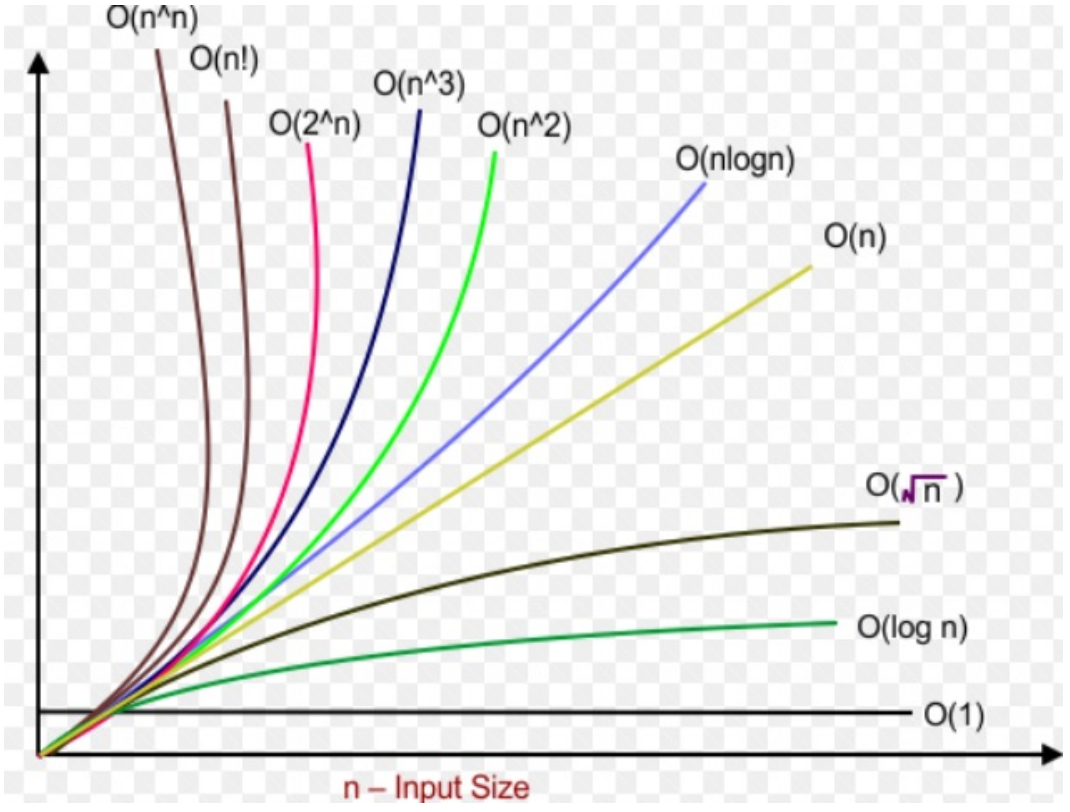
$\simeq O(m)$

Compare

	$O(n)$	$O(n^2)$	$O(n^3)$
$n = 1$	1	1	1
$n = 2$	2	4	8
$n = 3$	3	9	27
⋮	⋮	⋮	⋮
$n = 10^5$	10^5	10^{10}	10^{15}
	1 sec		10 min



$n = 10^5$



- ✓✓ $O(1)$ — 10 msec — constant
- ✓✓ $O(\log(n))$ — logarithmic
- $O(\sqrt{n})$
- ✓✓ $O(n)$ — linear
- $O(n \log(n))$
- ✓✓ $O(n^2)$ — quadratic
- ✓✓ $O(n^3)$ — cubic
- $O(2^n)$
- $O(n!)$
- $O(n^n)$ — 1 years

fastest ↓ slowest

Ex:-

```
for( int i=0; i<n; i+=2){
    Syso(⟶)
}
```

Operations = $n/2$

T.C = $O(n/2)$

$\cong O(n)$

ex:- $n*2 = O(n)$

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

$n-2 = O(n)$

$n/2 = O(n)$

$y = n/2$

$= O(n)$

$y = 5n^2 + 2n$

$O(n^2)$

op:- $2*n+5 = O(n)$

op:- $7*n^2 + n/2 + 27 = O(n^2)$