Time Complexity.

Algorithm.

↳ procedure / steps to solve a problem.

1 Problem

↳ more than 1 solution.

A1 A2

→ How much time?

→ Space.

Time & Space Complexity.

↳ To analyse our algorithm.

↳ save time ?

↳ save space

} both.

Time Complexity.

is it time taken by a process / algo to complete?

not in terms of times. sec / min / hours.

→ no. of i/p.

A [ ] = 

... n size.

i/p. milisecond. steps.

n

1	2	3	4	5

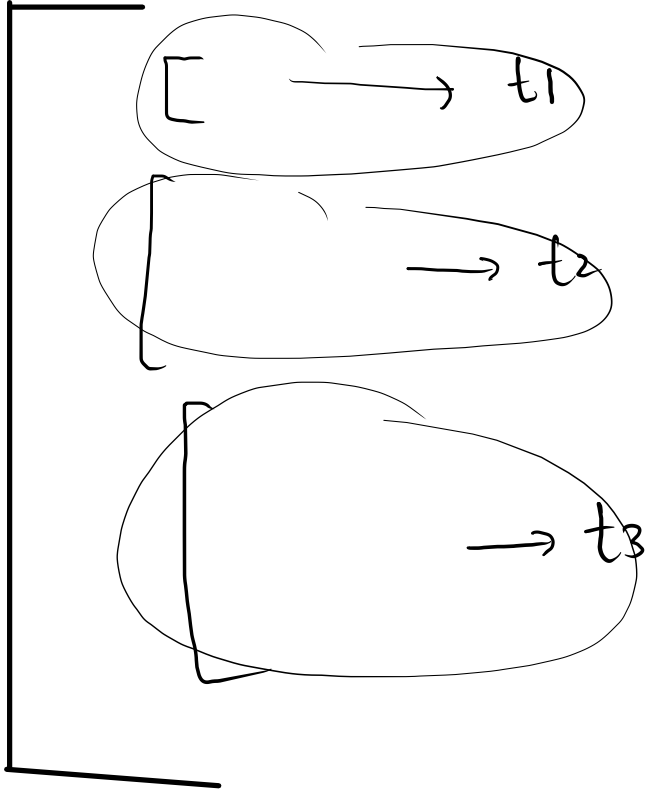
n size.

5 sec.

n sec.

Entire solution

$t1 + t2 + t3$



$\omega \rightarrow$ Best Case

$\Theta \rightarrow$ Avg. case

$O \rightarrow$ Worst Case. ☆

2	4	5	6	7	8	9	0	10

0

n size.

key: 2

20

70

18

$\Omega(1)$

Yes.

$\Theta($

2

0

$O(n)$

Binary Search

3

3

1	2	3	4	5	6	7

```
int a = 10;
x = scn.nextInt();
if (x == 10)
    break
    return
    c = a+b
s == c && b == 10
```

$O(1)$

Constant

↳ O(1)

a = scn.nextInt();

b

c

d

e

f

g

⋮

10

n = 100

n = 2

i/p

0 10

$$y^{2} + 2y + 6$$

order ?

2

i/p

max

1 →	a = 5w	5	1
1 →	b = 6w	3	2
1 →	c = 3w	2	3

3

O(1)

Que.

n — 500 — 3

5×10^5

5

1 2 3 4 5

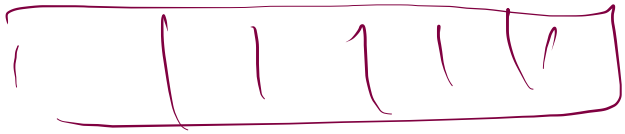
n more.

$O(n)$

n = 3

5 ✓ n

n = 500



$n = 5 \times 10^5$

```java
public class Main
{
    public static void main(String[] args) {

        int n = scn.nextInt();

        int a = scn.nextInt();

        System.out.println(a + n);


    }
}
```

3

$\rightarrow O(1)$

$\rightarrow O(1)$

$\rightarrow O(1)$

Constant $\rightarrow O(1)$

```java
public class Main
{
    public static void main(String[] args) {

        int n = scn.nextInt();

        for(int i = 0; i < n; i++){
            int a = scn.nextInt();
            System.out.println(a);
        }


```

1

//5000

//100

100 5000

n

$O(n)$.

```java
public class Main
{
    public static void main(String[] args) {

        int n = scn.nextInt();

        for(int i = 0; i < 10; i++){
            int a = scn.nextInt();
            System.out.println(a);
        }
```

1

10

10

11

11

const.

const.

T.C ?

n = 5000

n = 5

O(1)

Linear Search.

$O(n)$ → linear

1	2	3	4	5	7	6	8	9
0	1	2	3	4	5	6	7	8

Key = 7

n

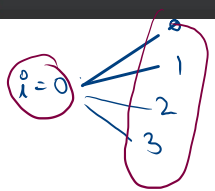
$O(n^2)$

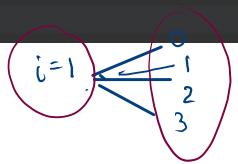
0	1	2	3
1	2	3	4

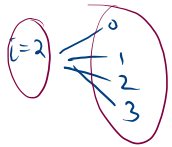
```java
int [] A =  {1,2,3,4};
int n = A.length;


for(int i = 0; i < n; i++){
    for(int j = 0; j < n; j++){
        System.out.println(A[i] + " " + A[j]);
    }
}
```

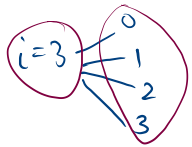
$O(1)$

$O(n^2)$









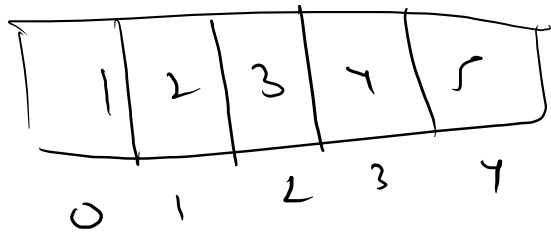
1 val of i → n times

n val of i → n × n times.

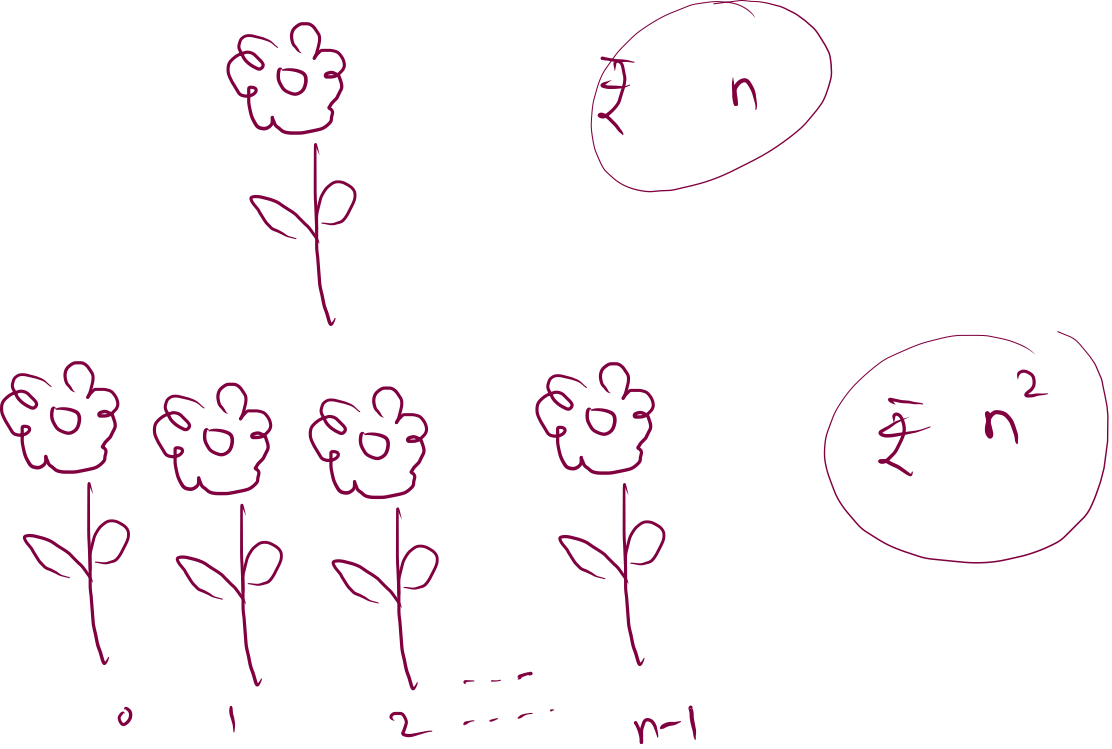
$O(n^2)$.

$n^2 \cancel{+1}$

```java
// int [] A =  {1,2,3,4,5};
int [] A =  new int[5];
A[0] = 1;
A[1] = 2;
A[2] = 3;
A[3] = 4;
A[4] = 5;
```



A.length.


n
n^2
0
1
2
n-1

n → i/p K → const.

```java
    public static void main(String[] args) {

        int [] A =  {1,2,3,4};
        int n = A.length;


        for(int i = 0; i < n; i++){
            System.out.println(A[i]);
        }

        for(int i = 0; i < n; i++){
            System.out.println(A[i]);
        }

    }
```

K

i/p

O(n)

O(n)

$O(2n + K)$

$O(2n)$

$O(n)$

$n + n \Rightarrow n$

$n * n \Rightarrow n^2$

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

$10n = n + n + n$ $O(n)$

1	7	3	2	4

O(n) → Nitish

100

O(n log n)

Sort

Fazila

1 2 3 4 7

TC → sort + 1

n=100

n log n

100 × 6 = 600

add item to your cart

less expensive.

```
for(int i = 0; i < n; i++){
    for(int j = 0; j < 10; j++){
        System.out.println(A[i]);
    }
}
```

const

$i = 0$ → 10 times.

$i = 1$ → 10 times

$i = 2$ → 10 times

$i = n-1$ → 10 times.

$n \times const$

$O(n)$

1 val i → 10

n values i → $10n$

$O(2n)$

$O(10n)$

$O(n)$

m , n

for (i=0 ——— i<m)

{ for (j=0 ——— j<n)

{

}

m >>>> n.

O(m).

m ≈ n

O(m×n) }

O(n×m) ✓

n×n

$\Theta(n^2)$

n → m

$0 \le n \le 10^5$ → 10^6 ✓

$0 \le m \le 10^5$

$n = 10^5$

$m = 10^5$

n ≈ m

$m = 10^6$

$n = 10^6$

```java
public static void main(String[] args) {

    int [] A =  {1,2,3,4};
    int n = A.length;


    for(int i = 0; i < n; i++){
        for(int j = 0; j < 10; j++){
            System.out.println(A[i]);
        }
    }
```

$10 \times n$

TC

$\rightarrow \Theta(n)$

```java
{
    public static void main(String[] args) {

        int [] A =  {1,2,3,4};
        int n = i/p;


        for(int i = 0; i < 5; i++){
            for(int j = 0; j < 10; j++){
                System.out.println(A[i]);
            }
        }


    }
}
```

1

2

5 x 10

x 10

5 3

O(1).

1000.

999.

n = 567 → 56 → 5 → 0

d = ~~0~~ 5

ans = ~~0~~ ~~7~~ ~~76~~ 765

```
int ans = 0;
while(n > 0){
    int d = n % 10;
    ans = ans * 10 + d;
    n = n/10;
}
System.out.println(ans);
}
```

1.
2.
3.

3 times.

O(1).

56770

5670

570

0700.

n = digit.

500

1 2 1 2 1 5 7 6 7 0 9 3 1 1 5 7

reverse

n

O(n).

```java
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();//5000

        int ans = 1;  -> O(1)

        int i = 0;  -> 1
        while(i < n){
            ans *= 10;
            i++;
        }

        System.out.println(ans);  1
    }
}
```

n

//5

//10^5

O(n)