

## Time Complexity - 2

Ex  $10^4$  numbers, **sort** the numbers

A (A's algo)

Macbook Pro

15 sec \*



15 sec

(Python)

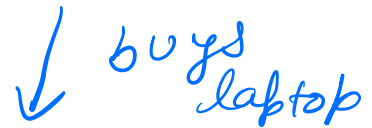
↓ C++

5 sec \*

B (B's algo)

Windows XP

20 sec



10 sec \*

C++



10 sec

# Execution time is not a good factor to compare algorithms.

Reason: Depends On H/W, lang, other factors.

for (  $i=0$  ;  $i < n$  ;  $i++$  )

do print (i)

$i \in [0, N-1]$  runs  $N$  times  
 $O(N)$

# Number of iterations is a good way of comparing algorithms.

Num of iterations =  $N-1, N-2, N-100$   
 $O(N)$

	Ar300	Raj
Number of iterations	$100 * \log_2 N$ ***	$\frac{N}{10}$

For smaller numbers, say  $N=20$

$100 \log_2 20$

$\frac{20}{10}$

400

2 \*\*

Do we need to prioritize smaller  $N$  or larger  $N$ .

Hotstar :  $IND$  vs  $NZ \approx 3$  crore

Youtube : Despacito  $\approx 7$  billion

Thus, we have to prioritize dealing with large numbers

# Asymptotic Analysis

↳ performance of your algorithm for very large input

→ Big O  
→ Theta  
→ Omega

for  $i=0; i < n; i++$

for  $i$  in  
range  $0, n$

# Calculate Big O from number of iterations.

- 1) Find the expression for number of iterations.
- 2) Ignore lower order terms.
- 3) Ignore constant co-efficient.

$$\frac{N}{10} = \left( \frac{1}{10} \right) \times N$$

↪ constant  
coefficient

Q  $N^2 + 4N$

$$N^2 + 4N$$

X

$$N^2$$

$$O(N^2)$$

Take  $N = 10^9$

$$N^2 + 4N$$

$$(10^9)^2 + 4 \times 10^9$$
$$10^{18} + 4 \times 10^9$$

# Contribution of smaller order term is insignificant.

## # Issues with Big O

1) Big O sometimes does not work for smaller values of  $N$ .

2)

A

B

$$10N^2 + 2N^{**}$$

$$11N^2 + 5N$$

$$O(N^2)$$

$$O(N^2)$$

# Both algo are  $O(N^2)$ , we cant compare using Big O

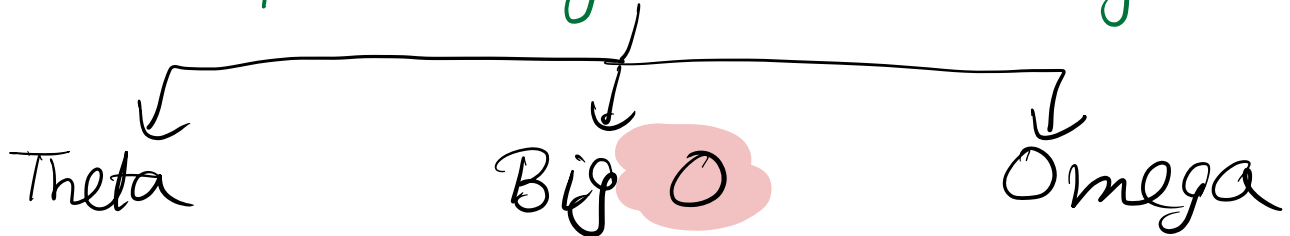
# For comparison in such cases, we will compare from number of iterations.

What is time complexity for this algorithm?

Time complexity



Perform asymptotic analysis



```
Q    for (i = 0; i < N; i++)  
    {  
        for (j = N; j > 0; j = j/2)  
        {  
            print (i * j)  
        }  
    }
```

What is TC for this code.

$i$	$j$	
$0$	$\log n$	
$1$	$\log n$	
$\vdots$	$\vdots$	
$N-1$	$\log n$	

}  $n$  times

$n \log n$

Number of iterations  
 $= N \log N$

TC:  $O(N \log N)$



# SPACE COMPLEXITY  $\rightarrow$   
Amount of extra memory used by  
our code

void fun (int N) {

4B  $\leftarrow$  int  $x = N$

4B  $\leftarrow$  int  $y = x^2$

8B  $\leftarrow$  long  $z = x * y$

8B  $\leftarrow$  double  $pie = 3.14$

}

Extra space used = 24 B

As 24 is constant, SC =  $O(1)$

Q void fun (int N)

{

int arr[N];

...

[4B] [4B] [4B] ...

}

} N

$$\text{Total memory} = 4 * N$$

$$SC: O(N)$$

Q void fun (int N)

$\alpha$   
 $4 \leftarrow \text{int } x = N$   
 $4 \leftarrow \text{int } y = x * 2$   
 $4 \leftarrow \text{int } z = x * y$

$4N \leftarrow \text{int arr}[N]$

$N^2 \leftarrow \text{bool matrix}[N][N]$

$y$

$$\text{Total space used} = N^2 + 4N + 12$$

$$SC: O(N^2)$$

Q

void fun (int arr[], int N)

{ int sum = 0

for (int i = 0; i < n; i++)

{

sum = sum + arr[i]

}

return sum

}

TC:  $O(N)$

SC:  $2 \times 4$   
 $= 8B$   
 $= O(1)$

Q bool fun (int arr[], int N, int k)

```
for (int i = 0; i < N; i++)
```

```
    if (all(Ci) == K)
        return true;
}
```

```
return false;
}
```

TC:  $O(N)$

SC:  $O(1)$

# Time Limit Exceeded

Say 1 sec -

$> 1 \text{ sec} \rightarrow \text{TLE}$

In case not given,

assume Time Limit = 1 sec

90 min

A  $\xrightarrow[\text{test}]{\text{Uber}}$  Q1  $\rightarrow$  TLE  $\rightarrow$  TLE  
 $\rightarrow$  TLE

Q2  $\rightarrow$  correct

What to do if you get

TLE?

$\Rightarrow$  Optimize your algorithm  
(make your algo more efficient)

$O(n^2) \rightarrow$  TLE

$O(n) \rightarrow$  Accepted  
(Correct Answer)

## Doubts

$N/10$

for ( $i=0; i < N; i = i+10$ )

$\times$  print( $i$ )

$y$

0, 10, 20, 30, 40

TL = 1 sec

$\downarrow$   
     $10^8$

$N = 10^5$

$O(N^2)$

$10^{10}$

$\rightarrow O(N)$

$10^5$   
(correct)

for  $i=0; i < n; i++$

{

for ( $j=1; \underline{j < i}; j=j+2$ )

{

—

}

0

1

2

⋮

⋮

$n-1$

0

$\log(1)$

$\log(2)$

$\log(3)$

⋮

⋮

$\log(n-1)$

$\log(1) + \log(2)$

+ ...  $\log(n-1)$

$\leq \underbrace{\log(n-1) + \log(n-1) + \dots}_{\text{blue bracket}}$

$$(n-1) \log(n-1)$$

$$n \log(n-1) - (n-1)$$

$$n \log(n-1) \approx n \log(n)$$