#BBM

- What/why is big-O?
- Why neglecting the lower order terms & constants?
- Why no. of iterations?

Numan

Vaibhav Malviya

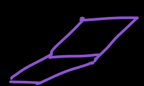
Solve Algo

Check For Algo

10s

15s

Hardware (Processor)



Mac book Pro

Samsung phone

10s

7s

Language

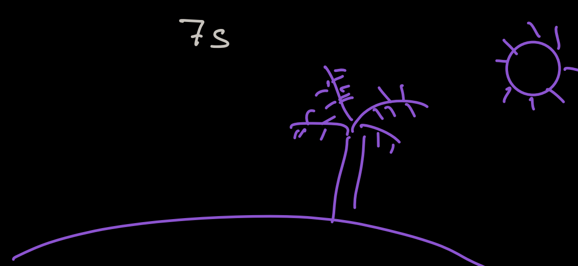
Python

C/C++

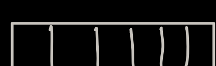
6.5s

7s

Physical condition



7.1s



6.9s

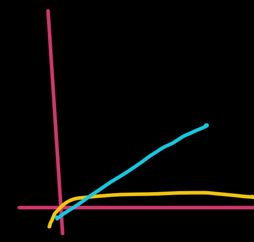
Input

5.3s

11s

iteration

$100 \log_{10} N$ $N/10$



if $N <= 3500 \longrightarrow$ Numan > Vaibhav

Vaibhav's algo to be preffered

$N > 3500 \longrightarrow$ Numan < Vaibhav

Numan's algo to be preffered

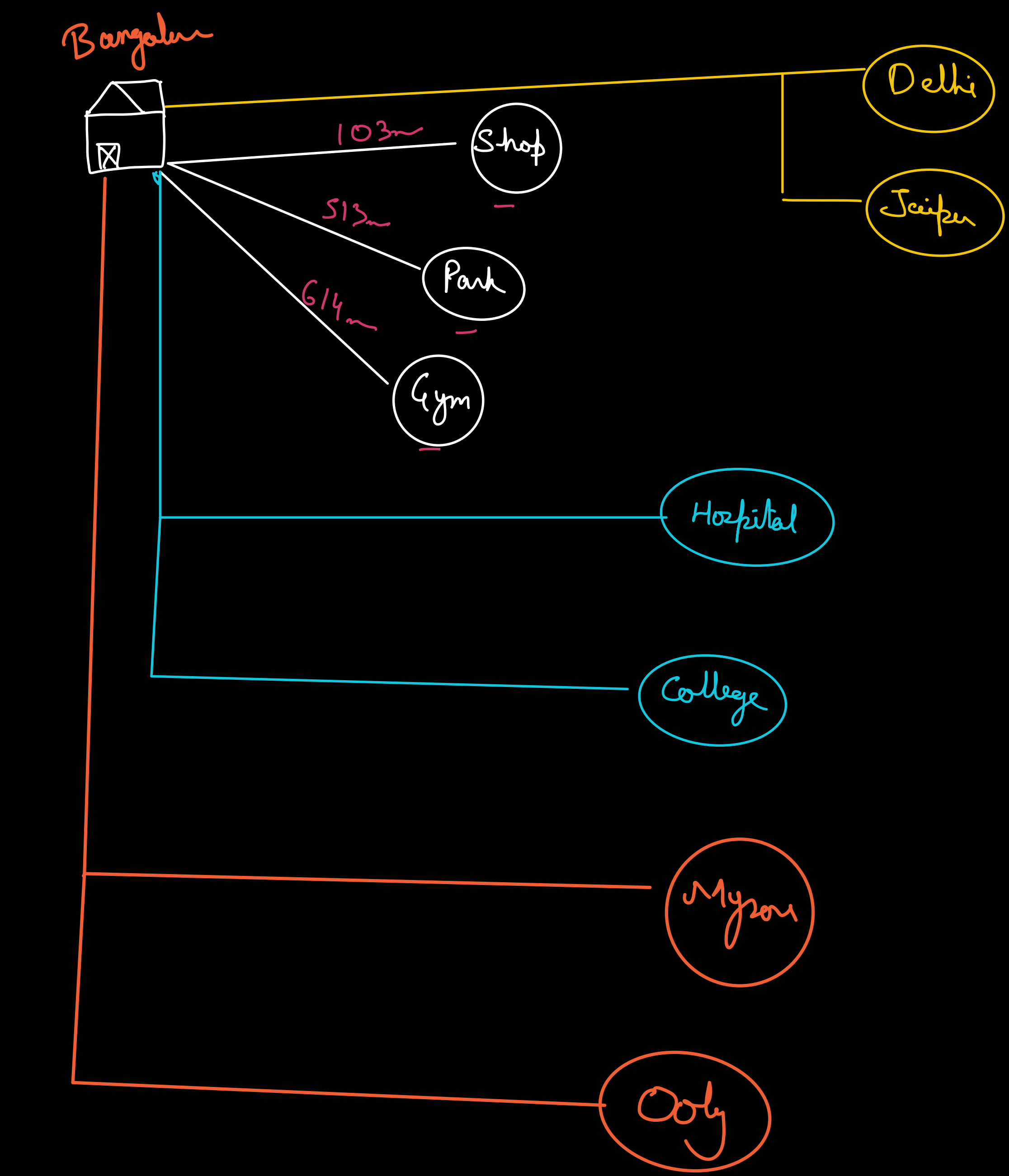
IPL $\longrightarrow$ 10 million

Google search $\longrightarrow$ millions

Baby Shark $\longrightarrow$ 10.84 Billion

Assymptotic Analysis

- Analyzing the performance for very large inputs.
- Big-O



Home ⟶ Moon   3, 84, 000 KM

Home $\xrightarrow[30\text{Km}]{}$ Airport $\xrightarrow[1500\text{Km}]{}$ USA $\xrightarrow[500\text{Km}]{}$ Space X Launch centre $\xrightarrow[3.84\text{ L Km}]{}$ Moon

Negligible

Algo $\longrightarrow$ No of iterations $N^2 + 10N$

Input size (N)	Total iteration	Contribution of lower order term (10N)	
$N = 10$	200	100	$\rightarrow$ 50%
$N = 100$	$10^4 + 10^3$ $= \underline{1}1000$	1000	$\approx$ 9%
$N = 10^4$	$10^8 + 10^5$	10^5	$\approx$ 0.1%

As input increases, contribution of lower order terms decrease.

Neglecting the constants

Edwin (Luka Algo)	Amit (your Code Algo)	
$10 \log_2 N$	N	Luka Algo
$100 \log_2 N$	N	Luka Algo
$1000 \log_2 N$	$N/10$	Luka Algo
$10N$	$N^2/100$	Luka Algo

$N = 100 \rightarrow 1000$ — 100

$1000 \rightarrow 1000$ — 10000

$10^4 \rightarrow 10^6$ — 10^6

$10^8 \rightarrow 10^9$ — 10^{14}

$N \log_2 N$	$100N$	Your Code

$N = 2^{100}$

$\boxed{2^{100} \times 100 \qquad 100 \times 2^{100}}$

$N = 2^{1000}$

$2^{1000} \times 1000 \qquad 2^{1000} \times 100$

N^3	$100N^2$	Your Code

$N=10 \longrightarrow 10^3$ — 10^4

$N=100 \longrightarrow 10^6$ — 10^6

$N=1000 \longrightarrow 10^9$ — 10^8

$N = \underline{100}$

$N^3 \Rightarrow N \times N \times N$

$100 \times 100 \times 100$

10^6

$O(1) < O(\log N) < O(\sqrt{N})$

↑
Constant
TC

$< O(N)$ ← Linear

$< O(N \log N)$

$< O(N^2)$ ← Quadratic

$< O(N^3)$ ← Cubic

⋮

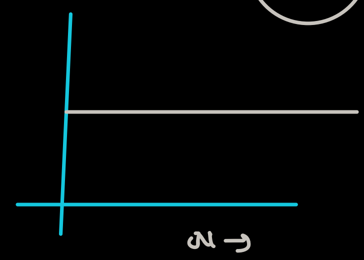
$< O(2^N)$ ← Exponential

$< O(N!)$ ← factorial

```
int add(int a, int b){
    ret a+b;
}
```

2 operations

$2 \times N^0 \Rightarrow O(1)$



Limitation of TC (Big-O)

	Algo 1	Algo 2	
	$10^3 N$	N^2	
TC :	$O(N)$	$O(N^2)$	
$N = 10$	10^4	10^2	$\rightarrow$ Algo 2
$N = 100$	10^5	10^4	$\rightarrow$ Algo 2
$\boxed{N = 10^3}$	10^6	10^6	$\rightarrow$ Both
$N = 10^3 + 1$	$10^3(10^3+1)$	$(10^3+1)(10^3+1)$	$\rightarrow$ Algo 1
			⋮

	$O(1)$	$O(N)$
	1000	N
$N = 10$	1000	10
$N = 100$	1000	10
$N = 1000$	1000	1000
$N = 10^4$	1000	10^4
$N = 10^9$	1000	10^9

$O(N^2)$

$2N^2 + 4N$

$N = 1000$ $\quad 2 \times 10^6 + 4 \times 1000$

$O(N^2)$

$3N^2$

3×10^6

$= 2 \times 10^6 + 10^6$

$O(N^3)$

$5N^3 + 6N^2$

$O(N^3)$

$4N^3 + 10N$

Input

```
bool search(int[] A, int K){
    for(i=0; i< A.length; i++){
        if (A[i] == K){
            ret true;
        }
    }
    ret false;
}
```

Iterations (N)

Best: 1

Worst: N

TC: $O(N)$

A : 8, 7, 6, 1, 2

K = 8

K = 10

Space Complexity

int ⟶ 4 Bytes / C_1 Bytes

long ⟶ 8 bytes / C_2 Bytes

```
fn (int N) {
    int x = N;       ⟶ 4B
    int y = 2*x;     ⟶ 4B
    int z = y+1;     ⟶ 4B
}
```

$\Rightarrow 12B \times N^0$

↓

$O(1)$

Constant

```
void fn(int N){
    int x = N;              ⟶ 4B
    int y = x^2;            ⟶ 4B
    long z = y^3;           ⟶ 8B
    int[] A = new int [N];  ⟶ 4N Bytes
}
```

$16B + (4 \times N)B$

↓

$(4 \times N)B$

↘ $O(N)$

```
fn(int N){
    int[] A = new int[N];      ⟶ (4×N)B
    int[] B = new int[2N];     ⟶ (4×2N)B
}
```

$$(4N + 8N)\,B$$

$$12N \text{ Byte}$$

$$\downarrow$$

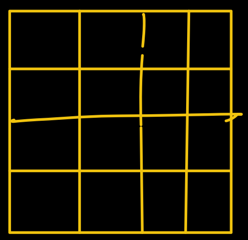
$$O(N)$$

```
fn(N){
    int[] A = new int[N];        ⟶ (4×N) Byte
    int[][] B = new int[N][N];   ⟶ (4×N²) Byte
}
```

$$(4N + 4N^2) \text{ Byte}$$

$$\downarrow$$

$$O(N^2)$$



$4 \times 4 \Rightarrow 16$

Input → { Algo
Computation } → Output

Space complexity
=
input Space + Extra/Auxiliary Space + Output Space
x — Created in program — x

Solve the problem without any extra space ?
→ O(1) SC
Constant space

```
int s = 0;              → 4B
int a = 10;             → 4B
long b = 10000;         → 8B
for (i = 0; i < N; i++){    (i → 4B)
        s = s + i;
}
```

20B → O(1)

```
int[] A = new int[100];
int[] A = new int[26];   → O(1)
```

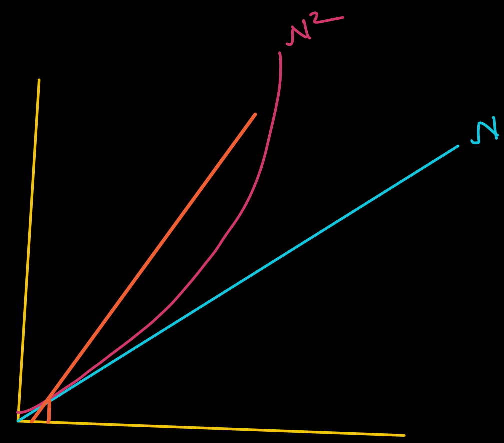
26 → NX

$$\cancel{N} \times \underline{\log N} < \cancel{N} \times \underline{N}$$

```
Print(1);
Print(2);
Print(3);
   ⋮
Print(1000);
```

$\longrightarrow$

```
for(i=1; i<=1000; i++){
    Print(i);
}
```



$O(1)$

```
int[] fn(int N){
    int[] A = new int[N];   → O(N)
       ⋮
    ret A;
}
```