Big (O) –

1) Calulate iterations

2) Remove lower order terms

3) Remove constant coefficients

$f(N) = 10N^2 + \underbrace{5N + 4}_{\text{lower}}$

$\Rightarrow O(N^2)$

$f(N) = \underbrace{N\log N + N}_{\text{lower order}} + N^2$

// Task: Given 10^4 Integers and asked them sort in increasing

Data = {7, 1, 2, 4, -3, 5} ⇒ {-3, 1, 2, 4, 5, 7}

	Tanmay (Algo Tan)	Neha (Scalar)	
Execution Time:	15 sec	10 sec	} Neha is better
→	Windos 95	MAC book pro M1	
→	Mackbook (7 sec) C++	Macbook (10 sec) python	} Tanmay is better {C++ is faster python}
→	C++ (7 sec) (Hot Mountain)	C++ (5 sec) (Canada)	}

1) Execution is depending lot of External factor

2) What can be a proper Comparison factor

```
i=1; i<=N; i++){      N iterations → independent of External factors.
    print(i)
}
```

→ Iterations are Used } N - Input size

Tanmay	Neha
→ $100 \log N$	$N/10$

✓ → Tanmay   ✓ → Neha

N	Iterations	less time
<= 3500 }	Tanmay > Neha	Neha will take less time
> 3500 }	Neha > Tanmay	Tanmay will take less time

In General N Values are Very large

→ Since N very large Tanmay is better

Q1

Meenakshi	Anand	Rajesh	Neha
f(N)	g(N)	h(N)	$f_2(N)$

Assymptotic Analysis of Algorithms

{Big O}

Omega Ω

Theta Θ

TODO

observing performance of Algo for very large Input Value of N

⇒ Big (O)

→ Calculate iterations

→ Neglecting lower order terms

→ Neglecting constants

→ Big (O) holds after certain Threshold

Neglect lower order

$$f(N) = N^2 + 10N$$

$N = 10$

Total = 200

N^2	$10N$	Contr lower order in Total
100	100	$\frac{100}{200} * 100\%$ $\Rightarrow 50\%$

$N = 100$

Total = $10^4 + 10^3$

N^2	$10N$	lower %
10^4	10^3	$\left[\frac{10^3}{10^4+10^3}\right] * 100$

$\approx \frac{10^5}{10^4+10^3} \approx 10\%$

$N = 10^4$

Total = $10^8 + 10^5$

N^2	$10N$	% lower order
10^8	10^5	$\left[\frac{10^5}{10^8+10^5}\right] * 100$

$\Rightarrow \frac{10^7}{10^8+10^5} \approx 0.1\%$

If $N \uparrow$, contribution of lower order term decreases. That's why we neglect lower order terms.

// Neglect constant coefficient Terms

Task

Algo1	Algo2	Comp	
N^2	$10N$	$N^{\cancel{2}}$	$10\cancel{N}$
		N	10
N^2	$100N$	$N >$	100

Task2

Algo 1	Algo2
$5N\sqrt{N}$	N^2

$5\cancel{N}\sqrt{N} \leq N^{\cancel{2}}$

$5\sqrt{N}$ N

5 $\sqrt{N}$

$\sqrt{N} > 5 \} \; N > 25$

For large $N\sqrt{N}$

Ens:

$100\cancel{N}\sqrt{N}$ $N^{\cancel{2}}$

$100\sqrt{N}$ N

$\{100$ $\sqrt{N}\}$ $\sqrt{N} > 100 \}$ Apply Square

$N > 10^4$

// Issuses in Big(O) Notation

Task	Algo1	Algo2
	100N	N^2
→	$O(N)$ >	$O(N^2)$

} Can we say Algo1 is always better NO

N	Algo1 (100N)	Algo2 (N2)	
10	10^3	10^2	} Algo2 is better
50	5000	2500	} Algo2 is better
100	10^4	10^4	} Both Same
>100 (101)	(100)(101)	(101)(101)	} Algo1 is better!

N > 100 Algo1 is better

O(N) is better than O(N2) After a Certain Point

Threshold

// Issue which cannot resolved by Big (O)

Task:

Algo1 — Algo2

$\{10N^2 + 5N$ —— $5N^2 + 10^2 N\}$ Algo2 has better performance

→ $O(N^2)$ <——> $O(N^2)$

Spac Complexity

void fun (int N){ ← Input Valu

int ar[10] → 40 Bytes
float f; → 4 Bytes
double d; → 8 Bytes

}

Memory in Bytes

→ 52 Bytes : $O(1)$

→ Constant Space

void fun (int N){ ← Input Valu

int ar[10] → 40 B
float f; → 4 B
int ar[N] → 4N

}

Memory →

4N + 44B : $O(N)$

```
void fun(int N){
    int ar[10]          -> 40
    int ar1[N]          -> 4N
    int ar2[N][N]       -> 4N^2
}
```

Memory | Total

$40 + 4N + 4N^2$

$\Rightarrow O(N^2)$

// Given an array and sum of all Elements

```
int sum(int ar[], int N){
    s = 0;
    i = 0; i < N; i++){
        s = s + ar[i]
    }
    return s
}
```

4N + 8B

Input array

Algo space: 8B

$\rightarrow O(1)$

// Space Complexity: Based on Extra space we use to solve problem

Anything other than given input data Extra space

10:20PM

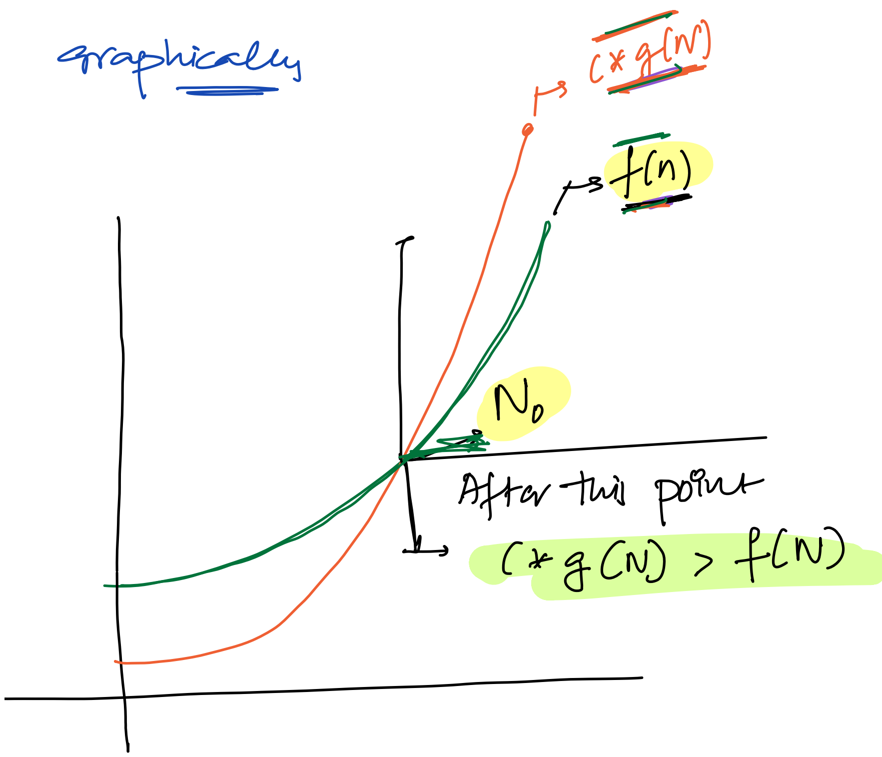
Mathematical Big (O) Representation

Given $f(n)$, it is $O(g(N))$ if there exists

2 constants C & N_0 such that

$\forall\ \boxed{N >= N_0}\quad \boxed{C * g(N)} > f(N)$

graphically



$$F(N) = O(g(N))$$

$$F(N) <= C * g(N) \quad \forall N >= N_0$$

$$F(N) = N^2 + N : O(N^2)$$

$$N^2 + N <= N^2 + N^2$$

$$\underbrace{N^2 + N}_{F(N)} <= \underbrace{2}_{C}\underbrace{N^2}_{g(N)} \} \quad \forall N >= \underbrace{1}_{N_0}$$

$$F(N) = N^2 + 5 \Rightarrow O(N^2)$$

$$N^2 + 5 <= N^2 + N^2 : O(N^2)$$

$$\underbrace{N^2 + 5}_{f(N)} <= \underbrace{2}_{C}\underbrace{N^2}_{g(N)} \} \quad N >= \underbrace{3}_{N_0}$$

$N = 1$ *	$N = 2$ *	$N = 3$ ✓ N
$6 <= 2$	$9 <= 8$	$14 <= 18$

$F(N) = 10N^3 + 10^6 : O(N^3)$ $\quad 11N^3 >= 10N^3 + 10^6$

$10N^3 + 10^6 <= 11N^3$ } $\quad N^3 >= 10^6$ } $\quad N >= 10^2$

$10N^3 + 10^6 <= 11N^3$ } $\forall N >= 10^2$

$f(N) \quad c \quad g(N) \quad N_0$

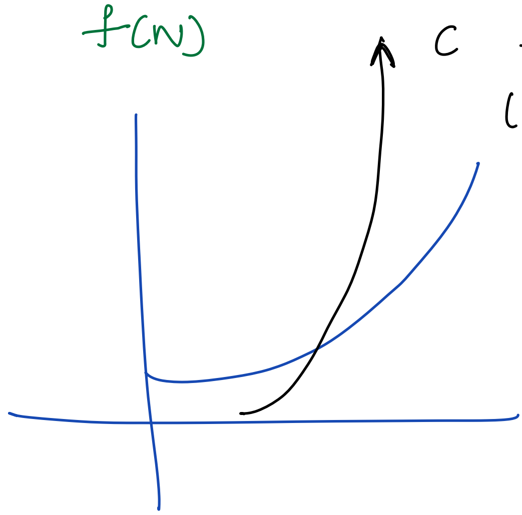
Tighter upper bound

$F(N) = N^2 + 10N + 8 : O(N^2) \quad O(N^3) \quad O(N^4) \quad O(N^5)$

$N^2 + 10N + 8 <= N^3 + 10N^3 + 8N^3$

$N^2 + 10N + 8 <= 19N^3, \quad N >= 1$

$f(N) \quad C \quad (N) \quad N_0$



graphically