⇒ 10^4 integers —Algorithm→ To sort data on Pc

Ex: ar[5] = {-3, 0, 5, -2, 1} ⇒ -3 -2 0 1 6

Shubham (Scaler sort) — Vishal (ProSort)

Execution Time :

- 15 sec (Windows Vista) → Macbook Pro
- 10 sec (Macbook Pro)
- 8 sec (C++)
- 10 sec (Python) → C++ → 7 sec (Mt Everest)
- 8 sec (C Volcano) → Mt-Everest

// Execution time not a fair to compare 2 Algos

//

```
for(i = 1; i <= N; i++){
|
}
```

→ Iterations : [N iterations]

≡ // Iterations are used for comparison

If N Numbers

Shubham	Vishal
⇒ 100 log N	→ N/10

∼ - Shubham

∼ - Vishal

	Iterations	Code Optimization
N <= 3550	: Shumam > Vishal	Vishal
N > 3550	: Vishal > Shubham	Shubham

For Larger Inputs N Shubham is better

Asymptotic Analysis
- Bpg O : O
- Bpg Theta : Θ
- Bpg Omega : Ω (TODO)

↳ Observing Behaviour / Growth of Algorithms for very large Inputs

Bpg (O) :

⇒ Calculate Iterations ✓ ⇒ f

1) Neglect lower Order Terms
2) Neglect Constant Coefficient Terms
3) Bpg (O) After Certain Threshold

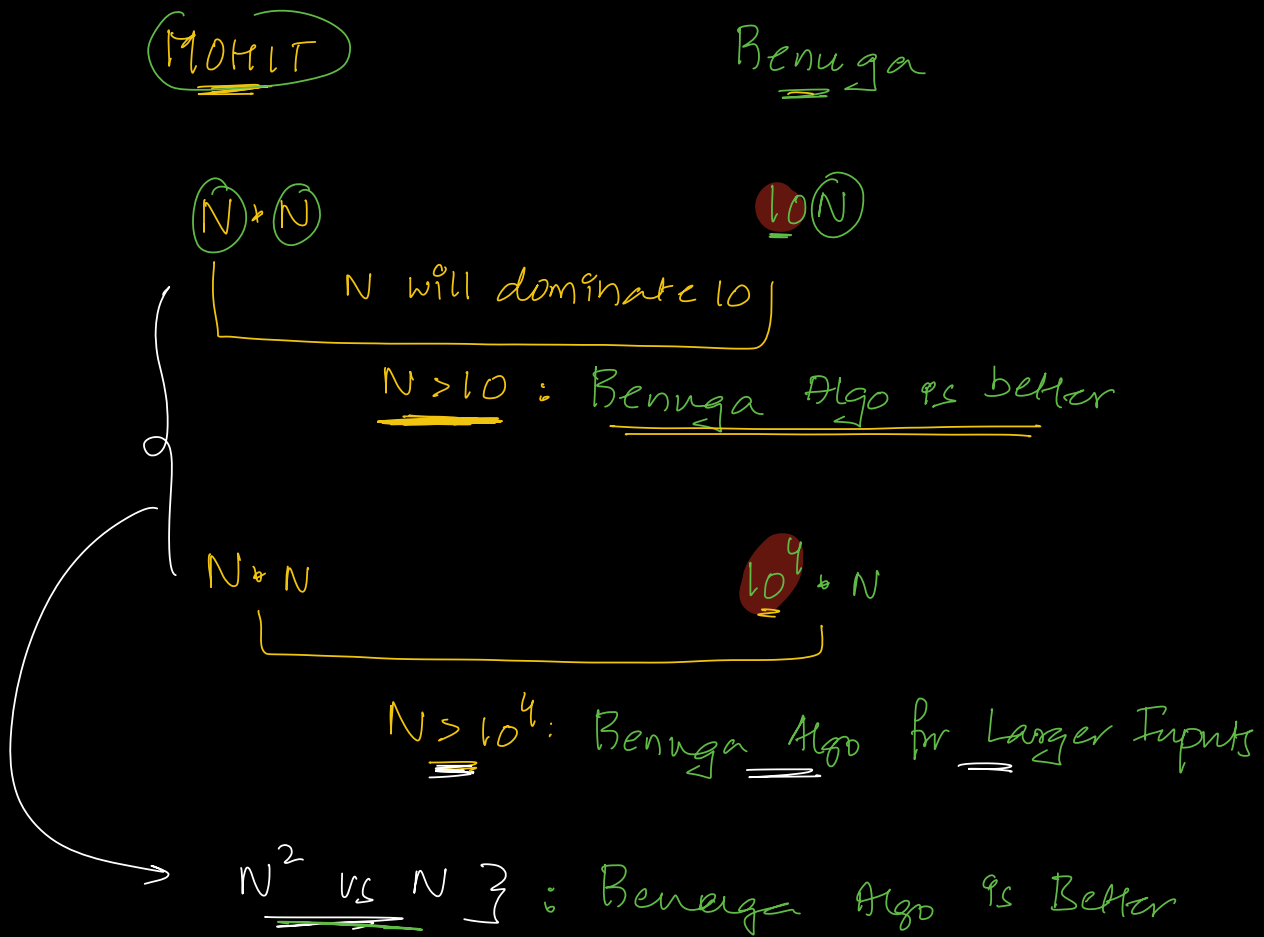
Arunvana : Autocomplete

↳ $N^2 + 10N$ Iterations

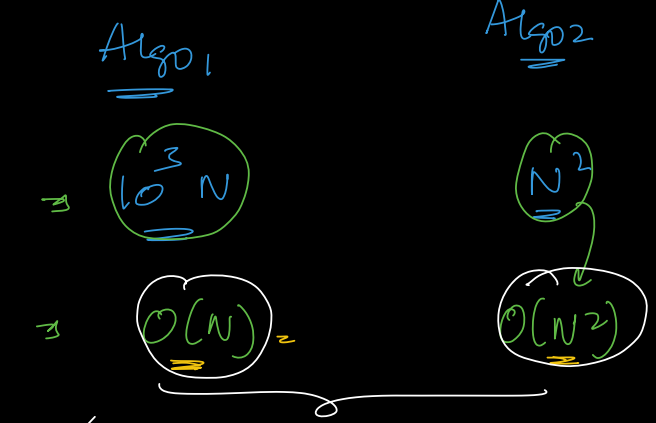
(N)	N^2	+	$10N$	=	Total	Contribution of $10N$ in Total Iterations
100 :	10^4	+	10^3	=	$10^4 + 10^3$	$\left\{\frac{10^3}{10^4+10^3}\right\} * 100\% \Rightarrow 9.09\%$
10^4 :	10^8	+	10^5	=	$10^8 + 10^5$	$\left\{\frac{10^5}{10^8+10^5}\right\} * 100\% = 0.1\%$

10^5 : $10^{10} + 10^6 = 10^{10} + 10^6$ $\left\{ \frac{10^6}{10^{10} + 10^6} \right\} * 100\% \approx 0.01\%$

// Recomender System



Issues with Big O



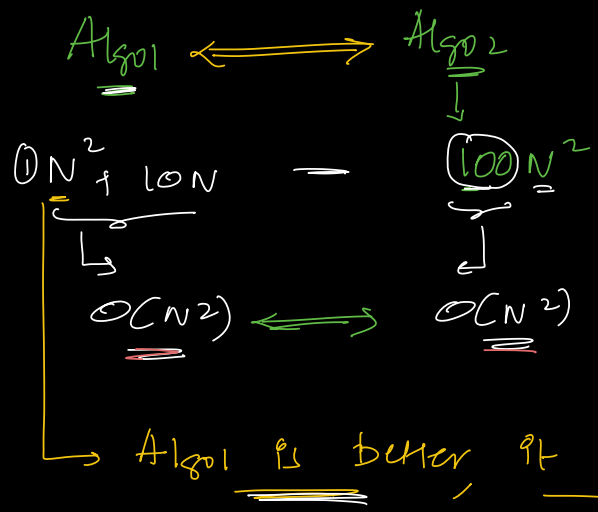
// Algo1 is always better x

N =	Algo1	Algo2	
	$10^3 N$	N^2	
10	10^4	10^2	} Algo2 is better
10^2	10^5	10^4	} Algo2 is better
10^3	10^6	10^6	} Both same Same
10^3+1	$(10^3)(10^3+1)$	$(10^3+1)(10^3+1)$	} Algo1 is better

Algo1 is better than Algo2 After some N

Threshold

//



// Space Complexity

int : 4B, long : 8B, double : 8B

```
void fun(N){
    int    a, b, c
    long   d
    double e
    print(e*d*b*c)
}
```

How much bytes = 28 Bytes

= O(1) Memory

```
void fun(N){
    int a, b, c
    long d
    double e
    int arr[N]
}
```

Total Bytes : 28 + 4N

↳ : O(N)

```
void fun(N){
    int a, b, c
    long d
    double e
    int arr[N]
    int mat[N][N]
}
```

Total Memory :

28 + 4N + 4N²

SC : O(N²)

```
for(i = 1; i <= N; i++){
    int x;
    // Scope of a variable lies
       with in { _ }
}
```

// Given N array elements calculate sum of all array elements?

N = 5 , {3, 7, 2, -1, 0} : Sum = 11

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

Space Complexity:

→ 4N + 12B → O(N) ×

// Input Space

// Extra Space we need to Solve Pr.. } Auxiliary Space

Extra Space = 8B

O(1)

⇒ O(1) //

```
void pref (int ar[], int N) {
    int sum[N];          ---> 4N
    sum[0] = ar[0];      ---> 4
    for (i = 1; i < N; i++) {
        sum[i] = sum[i-1] + ar[i]
    }
}
```

Input

4N + 4

SC: O(N)

```
void fun (int arr[], int N, int k) {
    for (i = 0; i < N; i++) {
        if (ar[i] == k) {
            return True
        }
    }
    return False
}
```

Best | Worst | {Avg}

Best: 1

Worst: N

D) Anirban

(Panic Algorithm)

Best: 1

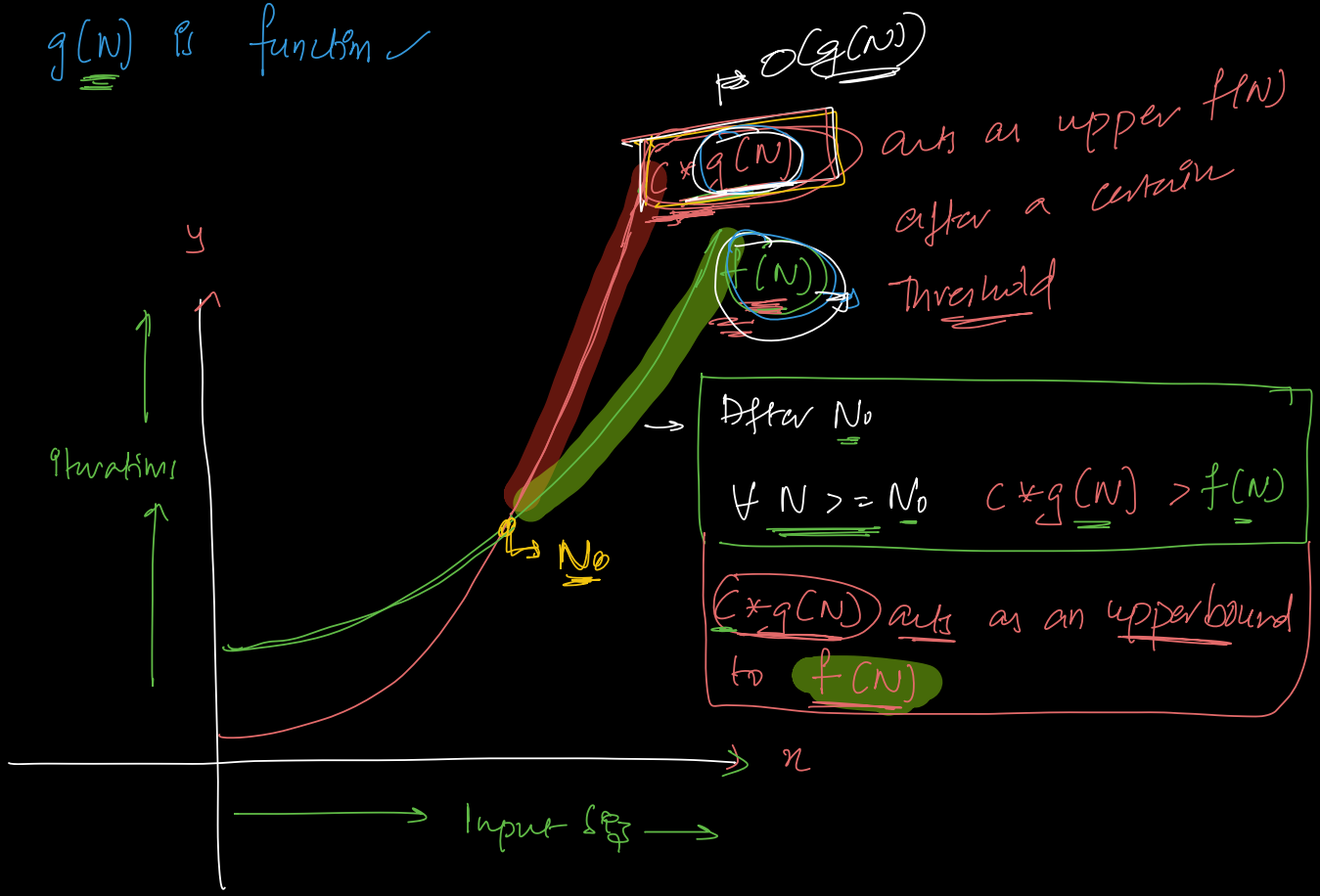
N^2 ---> $O(N^2)$

Big O: $f(N)$ it is $O(g(N))$ if there exists 2 constants (C & N_0) such that

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

$f(N)$ is function

$g(N)$ is function



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

$F(N) \propto = c * g(N) \quad \forall \; N >= N_0 = O(N^2)$ ✓

$F(N) = N^2 + 5N$

N	$N^2 + 5N$		$2N^2$
4	36 ✓	>	32
5	50	=	50
6	66	<	72
7	84	<	98

$\forall \; N >= 5$

$2N^2 >= N^2 + 5N$

$F(N) = N^2 + 5N$: ___

$N^2 + 5N \propto = 2N^2 \quad \forall \; N >= 5$

$F(N) \propto = c * g(N) \quad \forall \; N >= N_0 \rightarrow N_0 = 5$

$c = 2 \quad = g(N) = N^2$

$N^2 + 5N \quad \prec = \quad 10N^2 \quad \forall \; N_0 >= 0$

$F(N) \quad \prec = \quad c * g(N) \quad \forall \; N >= N_0$

$c = 10$; $g(N) = N^2$; $N_0 = 0$

$N^2 + 10N \quad \prec \quad 100N$ ✗ $O(N)$ ✗

$N = 100 : \quad 10^4 + 10^3 \quad > \quad 10^4$

$N > 101 : \quad (101)^2 + (10)(101) \quad > \quad (100)(101)$

$F(N) = 10N^3 + 10^6 \qquad 11N^3$

$N = 10 : \quad 10^4 + 10^6 \quad > \quad 11 * 10^3$

$N > 100 : \quad 10^7 + 10^6 \quad = \quad 11 * 10^6$

$N = 101 : \quad (10)(101)^3 + 10^6 \quad \prec \quad 11 * (101)^3$

$O(N^3)$

$10N^3 + 10^6 \quad \prec = \quad 11N^3 \quad \forall \; N >= 10^2$

$F(N) \quad \prec = \quad c * g(N) \quad \forall \; N >= N_0$

$c = 11$; $N_0 = 10^2$

$F(N) = 10N^2 : O(N^3)$ ✓ N^3 $O(N^4)$ $O(N^5)$

N	$10N^2$	N^3
10	1000	= 1000
11	$10 * 11^2$	< 11^3

$10N^2$ $\propto$ $= 1 \times N^3$ $\forall$ $N >= 10$

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

$C = 1$ $\quad$ $g(N) = N^3$

$F(N) \Rightarrow 100N^3$

- $O(N)$ ✗
- $O(N^2)$ ✓ closest upper bound
- $O(N^3)$ ✓
- $O(N^4)$ ✓

→ Running Speed slower than Speed of light ✓ upper

Running Speed slower than Train ✓ upper bou

slower than Bike ✓ upper bound

Slower than Bolt ✓ upper bound

Slower than 6km/hr ✓ upper bound

Tighter upper bound

// 1 = 1

Sum of 2 odd : $1+3 = 2^2$

Sum of 3 odd $1+3+5 = 3^2$

Sum of 4 odd $1+3+5+7 = 4^2$

Sum of 5 odd $1+3+5+7+9 = 5^2$

Sum of (n) odd → $(n)^2$

```
for (i = 1; i <= N; i = i+2) {
    for (j = 1; j <= i; j = j+1) {
        |
        |
    }
}
```

// How many odd [1 N]

→ (N+1)/2

i	j : [1 i]	Iterations, i = i+2
1	j : [1 1]	1
		+
3	j : [1 3]	3
		+
5	j : [1 5]	5
		+
7	j : [1 7]	7
		+
9	j : [1 9]	9
		+
		11

// Sum of All odd Numbers from [1 - N]

N = 9

[1+3+5+7+9]

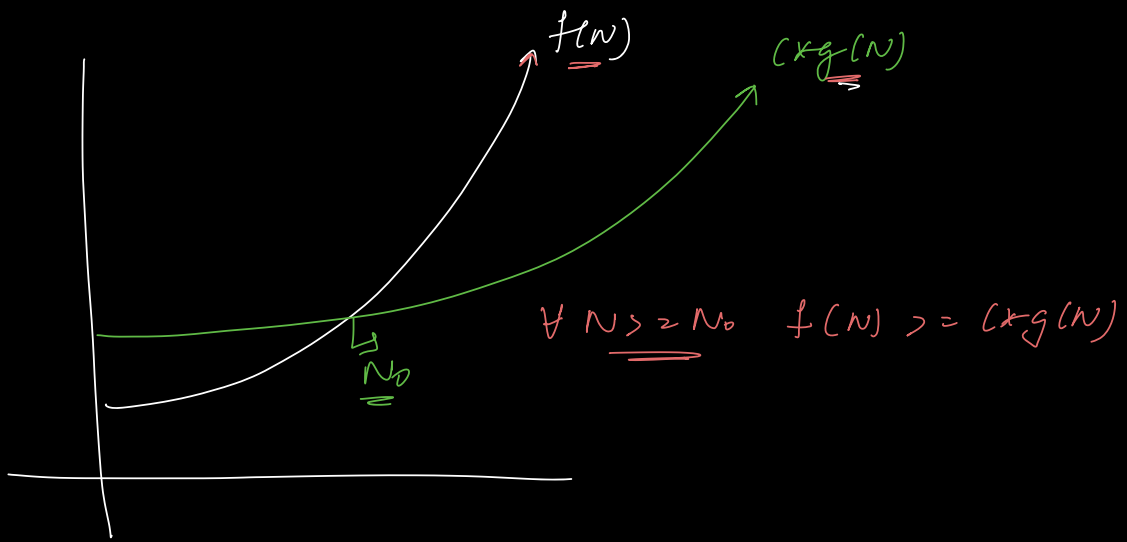
// Sum of first $\left(\frac{N+1}{2}\right)$ odd Numbers $= \left(\frac{(N+1)}{2}\right)^2$

$= \frac{N^2+2N+1}{4}$

// Omega $\Rightarrow$ (lower bound) { ——— }

$f(N) = \Omega(g(N))$ if there exists 2 constants C & N_0 such that

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



Sorting $\Rightarrow$

1) Merge Sort
2) Quick Sort
3) Radix Sort
3) Count Sort

$\Omega(N)$

// Theta = (Tight bound)

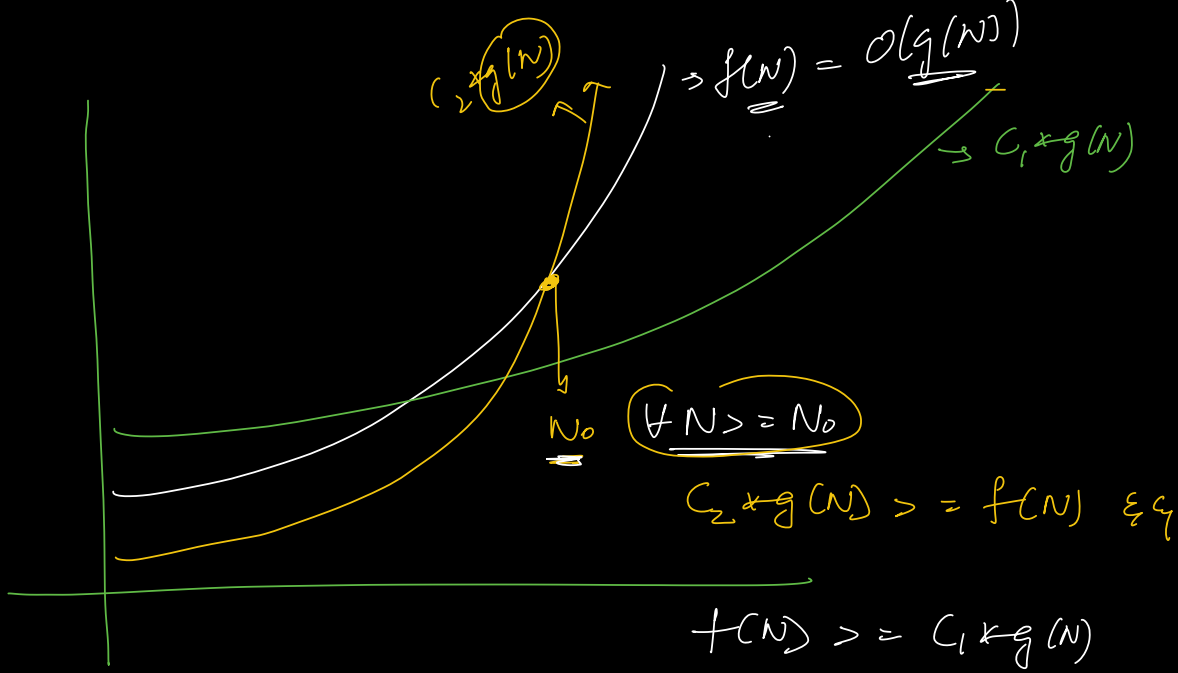
$\Rightarrow f(N) = \Theta(g(N))$ iff there exists 3 constant C_1 C_2 N_0 such that

Iterations

$C_1 * g(N) \quad \alpha = f(N) \quad \alpha = C_2 * g(N) \quad \forall N >= N_0$

lower bound — upperbound

$f(N) = 10N^2 \rightarrow \Theta(N^2)$



Merge Sort → $\Theta(N \log N)$

Beginner → Intermediate → Plenty of time

1) Leet code ×
2) Geeks vs
3) Hacker-rank ↦ W!S

→ N^2 ques → $\Omega(N^2)$ ✓

↳ $\Omega(N)$ ✓

↳ $\Theta(N^2)$ ✓