Execution time is a good measure to compare the algorithm.

Sorting championship

7	4	1	15	2

⇓

1	2	4	7	15

Brijesh
(swap sort)

10 s ✓

macbook bro

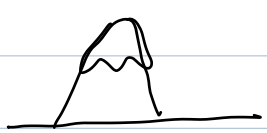
same machine

10 s

Python

same language

6 s ✓



Vaishali
(Alpha Sort)

15 s

samsung

same machine

7 s ✓

C++

same language

8 s

Execution time depends on a lot factors.

Physical factor

H/w factor.

```
for( i=1 ; i<=100 ; i++){
  |   =
}
```

100 iterations will run irrespective of the conditions.

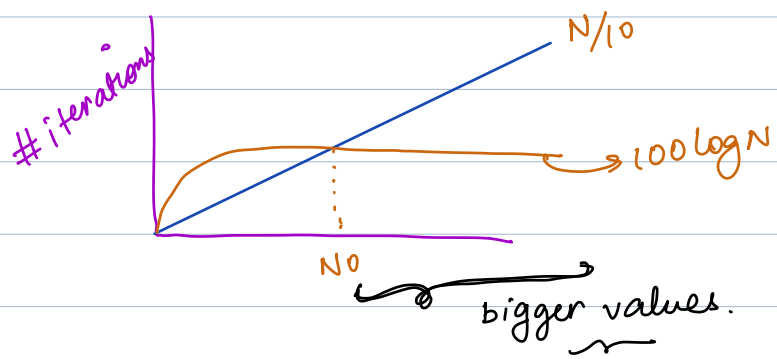
No. of Iterations

Brijesh

$100 \log N$

Vaishali

$\frac{N}{10}$



$N \leq 3891$ $\quad N/10$

$N > 3891$ $\quad 100 \log N$

Hotstar	Despacito	Pingu
2CR	2×10^9	7×10^9

Since data is growing rapidly → Always judge an algo based on large values of N

Above b/w $\frac{N}{10}$ & $100 \log N$

⇓

performed better

Asymptotic Analysis

Observing behavior / growth of algorithm w.r.t to large values of (N)

- Big O → worst case
- Big Θ → Average case
- Big Ω → Best case.

} Mathematical definition in advanced.

Steps of Big O

1. Calculating the no. of iterations
2. Neglect the lower order terms
3. Remove constant from the remaining term

#iterations

$4N^2 + 3N + 1 \rightarrow 4N^2 \rightarrow N^2 \rightarrow O(N^2)$

$100N^2 + 32N^3 + 5N + 1007 \rightarrow 32N^3 \rightarrow O(N^3)$

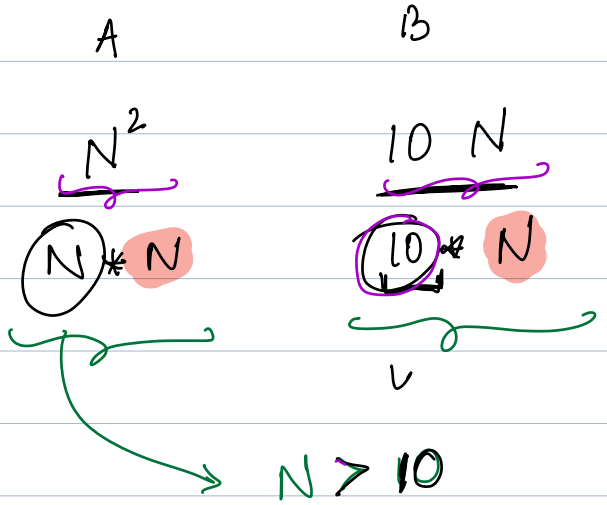
(2) Neglect lower order term.

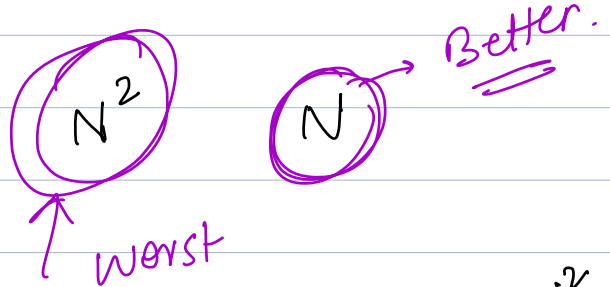
Home ——— 384000 km ———> Moon

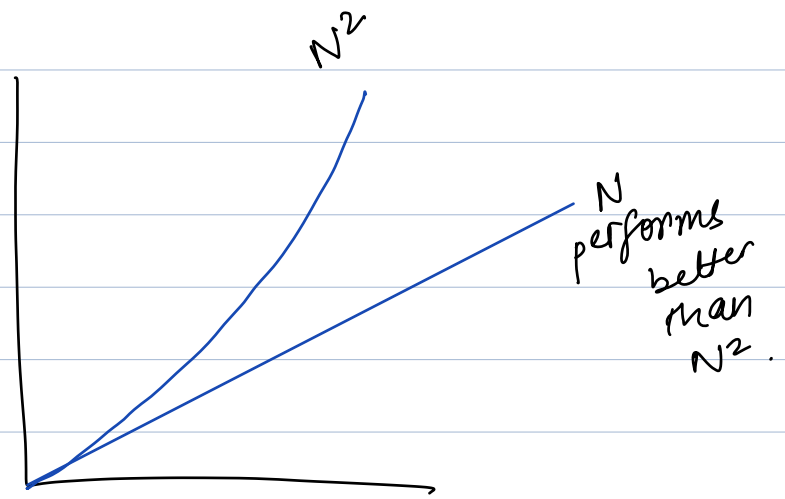
Home → Airport → US Airport → SpaceX → Moon

	$N^2 + 10N$	Contribution of lower order term
$N = 10^2$	$10^4 + 10^3$	$\frac{10^3}{10^4 + 10^3} \times 100 = 10\%$
$N = 10^4$	$10^8 + 10^5$	$\frac{10^5}{10^8 + 10^5} \times 100 = 0.1\%$
$N = 10^5$	$10^{10} + 10^6$	$\frac{10^6}{10^{10} + 10^6} \times 100 = 0.01\%$

(3) why to remove constant from higher order term.







Issues

Algo 1 Algo 2

(1)

$10^3 N$ $\quad$ N^2

better $\quad O(N) < O(N^2)$

N	$10^3 N$	N^2	
10	10^4	10^2	Algo 2 is better
10^2	10^5	10^4	Algo 2 is better
10^3	10^6	10^6	Both are same

10^3+1 $\quad$ $10^3(10^3+1)$ $\quad$ $(10^3+1)(10^3+1)$

10^3+2

Better.

more iterations

Algo 1 performs better.

N^2 $\quad$ $N \log N$

$N * N$ $\quad$ $N * \log N$

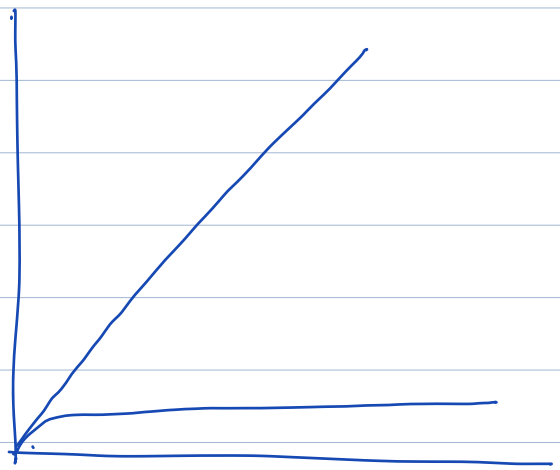
$N = 2^{32}$

$\log_a a^x$

2^{32}

$\log_2 2^{32}$

32



(2) #Iterations

Algo 1 [Better]

N^2 → Better

$O(N^2)$

Algo 2

$100N^2$ →

$O(N^2)$

We can compare iterations directly in such cases.

Amazon.

Break till 9:15

Cat in the air?

Billionaire

billie - in - air.

O(N)
will always be
better than $O(N^2)$

can N ever be
ve - ?
- NO!

Space Complexity

int ⇒ 4 Bytes
long ⇒ 8 Bytes
Double ⇒ 8 Bytes

```
void func(N){
    int a, b, c
    long d
    double f
    print(a+b+c+d+f)
}
```

$3 \times 4 + 1 \times 8 + 1 \times 8$

28 Bytes.

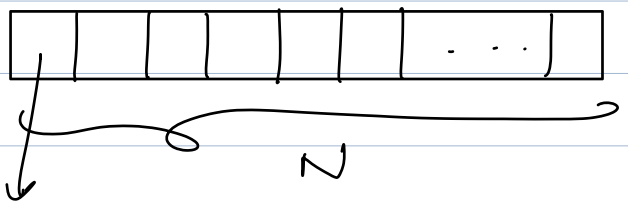
$O(1)$

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

$3 \times 4 + 1 \times 8 + 1 \times 8 + 4 \times N$

$= 28 + 4N$

$O(N)$



4 Bytes

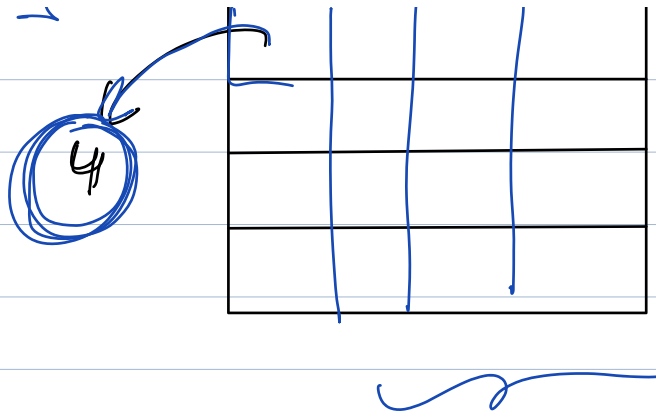
$4 \times N$

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

$3 \times 4 + 1 \times 8 + 1 \times 8 + 4 \times N + 4 \times N^2$

$28 + 4N + 4N^2$

$O(N^2)$



$N * N$

$4 \times N^2$

Scope of a variable lies within brackets.

```
for(int i=0; i<N; i++){
    int s;
    print(s)
}
```

i=0 → int s, print(s) — gets destroyed from memory.

i=1

Q. Given an array of size N, calc the

sum of all array ele.

3	7	2	-1	0

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

4×N + 4 + 4 + 4 → 8 Bytes

O(1)

[extra space & input space]

extra space: that you are taking

input space: you are given

```
int func(int arr[], int N){
    int sum[N]
    for(int i = 1; i<N; i++){
        sum[i] = sum[i-1] + arr[i]
    }
}
```

i/p given to us.

arr N sum i

}

~~4xN + 4~~ + 4xN + 4

SC: O(N)

Advance Problem → Find Big O

Assume power(,) → O(1)

```
func(N, K) {
    for(i = 1; i<=N; i++){
        int p = power(i, K)   // O(1)
        for(j = 1; j<=p; j++){
            print("yayyy")
        }
    }
}
```

SC O(1)

SC

i	j [1 ... i^K]	Iterations
1	[1 ... 1^K]	1^K
2	[1 ... 2^K]	+ 2^K
3	[1 ... 3^K]	+ 3^K
4	[1 ... 4^K]	+ 4^K
.		+
.		.

$N \left[1 \quad N^K \right] \quad + N^K$

$1^K + 2^K + 3^K + 4^K + \ldots N^K$

$K=1$ $\quad 1+2+3+4+\ldots N \quad \frac{N(N+1)}{2} \quad \frac{N^2}{2}$

$K=2$ $\quad 1^2+2^2+3^2+4^2+\ldots N^2 \quad \frac{N(N+1)(2N+1)}{6} \quad \frac{N^3}{3}$

$K=3$ $\quad 1^3+2^3+3^3+4^3+\ldots N^3 \quad \left(\frac{N(N+1)}{2}\right)^2 \quad \frac{N^4}{4}$

$\vdots$

$K \quad \frac{N^{K+1}}{K+1}$

$\frac{N^{K+1}}{K+1} \rightarrow O\left(\frac{N^{K+1}}{K}\right)$

$O\left(\frac{N^{K+1}}{K+1}\right)$