* We are going to do lot of close approximations.

Q. Given 10^4 no's, write an algo to sort these no's.

	Namit	Akshay
	Nam Algo	Aks algo.
Execution time :	* 10 sec	15 sec
	{Macbook Pro}	{Old windows}
		⇓ scaler
		{Macbook Pro}
	↓	↓
	10 sec (Python)	* 9 sec (C++)
	⇓	⇓
	C++ * 8 sec	9 sec
	⇓	⇓
	Antarctica (Cold)	Volcano (Hot)
		⇓
		Antarctica (Cold)
		* 7.5 sec

* Execution time is NOT a good factor to compare

depends on S/W, H/W & external factors.

* for(i = 1; i <= N; i++){

  - - - - -

  }

⇒ No. of iterations are NOT dependent on H/W, S/W & external factors.

Q. Given 10^4 no's, write an algo to sort these no's.

	Aman	Tanseer
# iterations	$100 \log_2 N$	$N/10$

Which algo is better.

$N = 2^{32}$

$\Rightarrow 100 \log_2 2^{32}$

$\Rightarrow 100 \times 32$

$\Rightarrow 3200$

$\frac{2^{32}}{10} \rightarrow \frac{2^{10} \cdot 2^{10} \cdot 2^{10} \cdot 2^2}{10}$

$1024 \simeq 10^3$

$\rightarrow \frac{10^3 \cdot 10^3 \cdot 10^3 \cdot 4}{10}$

$\rightarrow 4 \times 10^8$

if N <= 3900, Aman's algo is having more no. of iterations than Tanseer's algo.

N > 3900, Tanseer's algo is having more no. of iterations than Aman's algo.

⇒ Ind vs NZ, WC S.F 2019

3 Crore concurrent users.

⇒ 3×10^7 users

⇒ We'll see the no. of iterations, for very large input values.

Asymptotic analysis

⇒ Performance of your algorithm for very large input values.

$(N \rightarrow \infty)$

1) Big O ⇒ Worst Case
2) Omega
3) Theta

How to find the Big O notation :-

1) No. of iterations.
2) Neglect all the lower order terms.
3) Neglect all constant terms.

Ex Revanth's Algo.

$\#$ of iterations $= N^2 + 10N$

1) $N = 100$

$\#$ of iterations $= 100^2 + 10 \cdot 100$
$= 10^4 + 10^3$

% of 10N in total no. of iterations $= \dfrac{1000}{10^4 + 10^3} \times 100$
$\simeq 10\%$

2) $N = 10^5$

$\#$ of iterations $= (10^5)^2 + 10 \cdot 10^5$
$= 10^{10} + 10^6$

% of 10N in total no. of iterations $= \dfrac{10^6}{10^{10} + 10^6} \times 100$
$\simeq 0.01\%$

3) $N = 10^6$

$\%$ of $10N \ll 0.01\%$

⇒ NOTE: Contribution of lower order terms is very negligible w.r.t to higher order terms.

$\Rightarrow O(N^2)$

Ex

1) $10 \log N \Rightarrow O(\log N)$

2) $10^2 \log N \Rightarrow O(\log N)$

3) $10^4 \log N \Rightarrow O(\log N)$

4) $10^4 N + 10^6 \Rightarrow O(N)$

5) $10^3 N \log N \Rightarrow O(N \log N)$

6) $N + 10 \log N \Rightarrow O(N)$

⇒ Constants & lower order terms are NOT effecting much the # of iterations when N is very large.

Issues with Big O notations

1) Mohan's Algo — Mayur's Algo

	Mohan's Algo		Mayur's Algo
	$100N$		N^2
	$\Rightarrow O(N)$		$O(N^2)$
$N=10$	1000	>	100
$N=99$	9900	>	9801
$N=101$	10100	<	10201

$\Rightarrow$ $N >= 100$: Mohan's Algo is better

$N < 100$: Mayur's Algo is better

2) Arshad — Chinmay

Arshad	Chinmay
$10N^2 + 5N$	$11N^2 + 100N$
$\Rightarrow O(N^2)$	$\Rightarrow O(N^2)$

Both algo's are $O(N^2)$, we can't compare these 2 algo's based on their Big O notation.

Importance of Big O :

↳ makes comparisons very smooth.

Space Complexity

```
fun (int N) {
    int x = 10;      → 4B
    int y = x^2;     → 4B
    long z = x*y     → 8B
    double pie = 3.14;  → 8B
    ..........
}
```

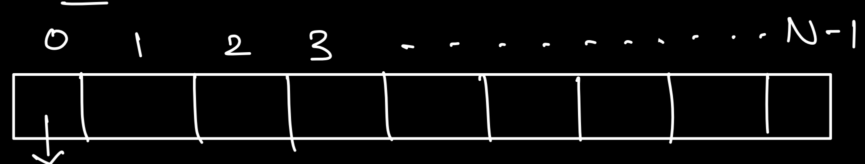
int ⇒ 4 Bytes
long ⇒ 8 B
double ⇒ 8 B.

Total memory = 24 Bytes.

SC : O(1) { Constant Extra Space }

Ex

```
fun (int N) {
    int A[N];
    - - - - -
}
```



int (4B)

⇒ N int ⇒ 4N

Extra Space = 4N
(Auxilliary Space)

⇒ O(N) ⇒ SC

Ex

```
fun (int N) {
    int x = N;              → 4B
    int y = 100;            → 4B
    long z = x*y;           → 8B
    double pie = 3.14;      → 8B
    int arr[N];             → 4N B
    bool mat[N][N];         → N^2 B
}
```

Extra Space: $24 + 4N + N^2$

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

Ex

```
int fun (int N) {
        int x;   // 100 variables
        ⋮
}
```

SC : 400 Bytes

: $O(1)$

Q: Given an Array, find the sum of array elements.

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

4B

4B

Extra space : 8B.

: SC : O(1)

⇒ We don't consider the input space in the Space Complexity

⇒ Amount of extra space created is called as S.C

Linear Search

```
bool fun( int arr[], int N, int K){
        for(i= 0; i< N; i++){
4B          if(arr[i] == K){
                return true;
            }
        }
        return false;
}
```

10	20	5	60	15	18

↑

K = 10 ⇒ iterations = 1

K = 18 ⇒ iterations = 6 (N)

TC { Worst Case : N, Best Case : 1 } O(N) {WC}

SC: O(1)

TLE : Time Limit Exceeded

↳ Code is taking more than expected time.

⇒ Reduce the no. of iterations to overcome TLE.

Polynomial : N^1 | N^2 | N^3 | - - -

Linear ⇓ (N^1), Quadratic ⇓ (N^2), Cubic (N^3)

Exponential :- 2^N | 3^N | . . . . .

Constant : $O(1)$