* Time and space complexity
* Asymptotic analysis
* Big O
* TLE - Time Limit exceeded.

Q Given 10^4 numbers, write an algorithm to arrange them in asc order.

	Aarthi	Nikhil
	Super Aarthi Algo	Nikhil Rocks Algo
execution time ⇒	15s	20s
	(Mac Pro)	(Windows 7).
		12s (Mac)
		(C++)
	(python)	
	10s (C++)	[Rajasthan]
	[Bangalore]	[Bangalore] 8s

*** When comparing algorithms, execution time is not a good factor :

it depends on multiple factors :

* S/W
* H/W
* external factors.

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

$\Rightarrow i \in [1, N]$

$\downarrow$

N iterations

never changes irrespective of S/W, H/W, other factors.

	Suyash	Arghya
iterations $\Rightarrow$	$100 \log_2 N$	$N/10$.

for $N <= 3500 \rightarrow$ Arghya's code is better

$N > 3500 \rightarrow$ Suyash's code is better

N		Winner
100	$\longrightarrow$	A
1000	$\longrightarrow$	A
10000	$\longrightarrow$	S.

$\Rightarrow$ Asymptotic analysis of algorithms :-

$\downarrow$

performance of your algos. for very large inputs.

Asymptotic notations :-

Big (O) $\longrightarrow$ will study & use.

Omega (Ω)

Theta (Θ)

$100 \log N \longrightarrow \log N$

$N/10 \longrightarrow N$

$\Big] \rightarrow$ easy to compare

why

i) neglect lower order terms

ii) neglect const terms.

Neha

iterations $\Rightarrow N^2 + 10N$.

$O \Rightarrow N^2$.

excluded $\rightarrow 10N$.

% of $10N \Rightarrow (N^2 + 10N)$.

$$= \frac{10N}{N^2 + 10N} \times 100.$$

$N = 100$

$$\%\ \text{exclusion} = \frac{10 * 100}{10000 + 1000} \times 100$$

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

$N = 10^5$

$$\%\ \text{exclusive} = \frac{10 \times 10^5}{(10^5)^2 + 10 \times 10^5} \times 100$$

$$= \frac{10^8}{10^{10} + 10^6} \approx \underline{0.01\%}$$

as, N inc. , the % of 10N dec.

So, we can exclude 10N for comparison

A	B
✓ $10\log_2 N$	N
✓ $10^2 \log_2 N$	N
✓ $10^4 \log_2 N$	N
✓ $10^4 N + 10^6$	N^2
N^2	✓ $10^4 N \log N$

Issue with Big O

	Bibin	Utsav
Iterations →	100N	N^2
Big O →	N	N^2

↓

efficient

Comparisons

		Bibin (100N)	Utsav (N^2)
N = 10	→	1000	100
N = 50	→	5000	2500
N = 100	→	10^4	10^4

N = 101 → 100 x 101   101 x 101

Conclusion

N < 100 → Utsav's code is better

N >= 100 → Bishu's code is better

Threshold point

* Big(O) comparison holds true after a certain pt. known as Threshold point

	Harsh	Jaivik
Iterations →	$10N^2 + 5N$	$11N^2 + 100N$
Big(O) →	N^2	N^2

Same

* At some scenario, exact comparison is not possible.

** TC depends on iterations, and iterations depends on input, so TC should always depends on input [mostly].

Space Complexity :-

```
fun(int N){

    int x = N
    int y = N^2
    long z = x + y
    double pi = 3.14

}
```

SC :- $O(1)$

int → 4B

long → 8B

double → 8B

⟹ Space is const, not dependent on N

total memory = 4 + 4 + 8 + 8

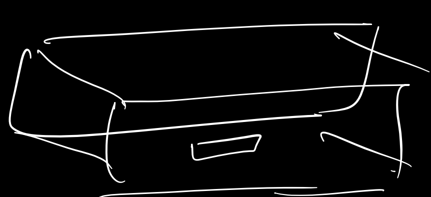
= 24B.

$N = 1, 100, 1000, 10^4, 10^5, 10^6$

int x = ___ ;

10^4 ← x

4B



```
fun (int N) {
    arr[N]  // creating an array of size N.
    for( i=0; i<N; i++) {
        print(i)

    int x = N
    int y = N+100
    int z = N+2N
}
```

space is not const, dependent on N.

$SC \Rightarrow O(N)$

$\Rightarrow$ find sum of all array element of size N.

```
void fun( arr[], N) {
    s=0
    for( i=0; i<N; i++) {
        s= s+arr[i]
    }
    print(s)
}
```

$$\boxed{\begin{array}{l} TC \Rightarrow O(N) \\ SC \Rightarrow O(1) \end{array}}$$

** Space complexity :- Amount of extra space taken by your algo. other than input data.

** use of any inbuilt func. like sort(), filter() etc should be added in TC & SC.

** use of any extra space → HashMap | Set | any DS should be considered in SC.

ex:

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

[1, 2, 6, 4, 11, 8, 7]   7   3

no. of iterations ⇒ k = 1 ⇒ 1

k = 7 ⇒ 7

k = 3 ⇒ 7

Best case ⇒ 1 iteration

worst case ⇒ N iteration ⇒ O(N)

↓

worst case iterations

° TLE ⇒ time limit exceeded :-

Rahul [Google].

↓

test (2 Qs. | 60 mins).

↓

Q1 ⟶ TLE ⟶ Optimise ⟶ TLE

↓ Optim

(✓) ⟵ TLE
Optim

59

* TLE ⇒ time limit exceeded

→ more time taken than ideal scenario

** TRICK will be discussed later

[Advanced]

Q 1.

```
func(N, k) {
    for(i=1; i<=N; i++) {
        p = pow(i, k)
        for (j=1; j<=p; j++) {
            print(j)
        }
    }
}
```

assum ⇒ pow() → O(1)

$TC :- O\left(\frac{N^{k+1}}{k+1}\right)$ $SC :- O(1)$

i	j	iteratio
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	$[1, N^k]$	N^k

total iterations

$= 1^k + 2^k + 3^k + 4^k + \dots + N^k$

$K=1; \Rightarrow 1+2+3+4+\dots+N = \frac{N(N+1)}{2} = \frac{N^2+N}{2}$

$\Rightarrow N^2/2$ $\Rightarrow O(N^2)$

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

$$= \frac{(N^2+N)(2N+1)}{6}$$

$$= \frac{2N^3+N^2+3N}{6} = O\left(\frac{N^3}{3}\right)$$

$$= N^3/3$$

$$K=3 \Rightarrow 1^3+2^3+3^3+4^3+\dots+N^3 = \left[\frac{N(N+1)}{2}\right]^2$$

$$= \left(\frac{N^2+N}{2}\right)^2 = O\left(\frac{N^4}{4}\right)$$

$$= \frac{N^4}{4}$$

$$k=1 \longrightarrow O(N^2)$$

$$k=2 \longrightarrow O(N^3)$$

$$k=3 \rightarrow O(N^4)$$

$$\Rightarrow \cancel{\boxed{TC \Rightarrow O(N^{k+1})}}$$

wrong

$$TC \Rightarrow O\left(\frac{N^{k+1}}{k+1}\right)$$

$\Rightarrow$ can't neglect $(k+1)$ as it is dependent on k, which is a user input

i	ibo
1	1
2	1
3	1

```
for ( ———— ) {
    print( )
}
```