Given some $\underline{N}$ numbers, write an algorithm to sort them.

sort ↓ Inc / Dec

Abhijeet (super abhijeet)	vs	Jilna (big bang)
15 sec (Windows XP)		10 sec (Mac M1)
↓ (Mac M1)		↓
12 sec (C++)		10 sec (Python)
↓		↓
12 sec (Volcano)		8 sec (C++) (Antartica)
↓ (Antartica)		↓
8 sec		8 sec

Execution Time

C > C++ > Java > Py

→ Execution time is not a good factor

⇓ → S/w + H/w + External Reasons.

```
for(i=0; i<N; i++){
    |
}
```

[0 ... N-1]

$\boxed{\text{No. of iterations} = N}$

Ques :-

Given N elements, sort N elements.

$N = 2^{32}$

$\log_2 2^{32} = 32$

2^{32}

	Nivedita	Karthik
No of iterations	$100 \log_2 N$	$N/10$
	$O(\log_2 N)$	$O(N)$

→ Karthik's Algo was better.

Till $N <= 3500$. Nivedita had more iterations

After $N > 3500$. Karthik has more iterations.

→ Nivedita's Algo is better.

Real World Scenario :-

Hotstar :- 3×10^8

$O(N^2)$ ← $10N^2 + 5N + \frac{6N}{5}$

$O(N^3)$ ← $\frac{N^2 + N^3}{4}$

O ω θ

Asymptotic Analysis of Algorithms :-

Input size $\to \infty$

↳ performance of your algorithm for very large input

Big(O) Notation for an Algorithm.

1) Calculate No. of iterations
2) Neglect lower order terms
3) Neglect const coefficients

$N^2 + \cancel{10N}$

N^2

Malay :- Input size → N

\# of iterations ⇒ $N^2 + \cancel{10N}$ ← lower order ⇒

	Total No. of iterations	% of 10 N in total iterations
$N = 100$	$10^4 + 10^3$	$\frac{10^3}{10^4 + 10^3} \times 100 \simeq 10\%$
$N = 10^5$	$10^{10} + 10^6$	$\frac{10^6}{10^{10} + 10^6} \times 100 = 0.01\%$

↓ ↓

$N = \ldots$?

// Sort N Numbers

	(Aakash)	Midhin
Iterations	$10 \log_2 N$	N
	$10^2 \log_2 N$	N
	$10^3 \log_2 N$	N
	$10^4 N + 10^6$	N^2
	Better	

Issues with Big(O)

Ishan	Sunny
$100N$	N^2
$\downarrow$	$\downarrow$
$\boxed{O(N)}$	$\boxed{O(N^2)}$

→ Ishan's Algorithm is efficient always.

N	I'shan	Sunny
	100N	N^2
	O(N)	$O(N^2)$
10 :-	1000	100 ↳ Sunny's efficient
99 :	9900	9801 ↳ Sunny's efficient.
100 :	10^4	10^4
101 :-	100 * 101	101 * 101

I'shan's efficient

(N > 100) :- Ishan's Algo will be better.

↓

Ishan's Algorithm is better after a certain value(100)

↓

Threshold point.

Big(O) comparison holds true after certain threshold.

Vikas

$10N^2 + 5N$

$\downarrow$

$O(N^2)$

Malay

$11N^2 + 5N$

$\downarrow$

$O(N^2)$

$10N^2 + N^2 + 5N$

$\boxed{8 \text{ Min}}$

Space Complexity :-

$\{$ int : 4B, long : 8B, double : 8B, bool : 1B $\}$

```
fun(int N) {
  4B   int x = N
  4B   int y = x^2
  8B   long s = x+y
  8B   double pie = 3.14
}
```

$\Rightarrow 4+4+8+8$

$\boxed{\Rightarrow 24B}$

$\Rightarrow O(1)$ ← space

```
fun(int N) {
  4B    int x = N
  4B    int y = x^2
  8B    long s = x+y
  8B    double pie = 3.14
  4*N   int arr[N];
}
```

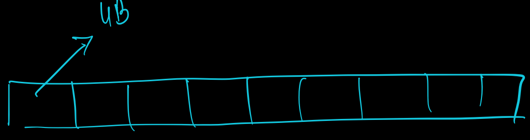
Total memory allocated in Bytes:-

$\Rightarrow 4+4+8+8+4N$

$\Rightarrow \cancel{24} + \cancel{4}N$

$\Downarrow$

$O(N)$



```
fun(int N) {
    int x = N
    int y = x^2
    long z = x+y
    double pie = 3.14
    int arr[N];
    int mat[N][N];
```

$$\underbrace{\text{int x, int y, long z, double pie, int arr[N]}}_{24+4N} \qquad \underbrace{\boxed{\text{int mat[N][N];}}}_{4N^2}$$

Total memory allocated in bytes:-

$$\Rightarrow \cancel{24} + \cancel{4}N + \cancel{4}N^2$$

$$\Downarrow$$

$$\cancel{N} + N^2$$

$$\Downarrow$$

$$O(N^2)$$

```
             X          x
             4N         4
fun(int arr[ ], int N){

4B    int sum = 0
                    int
4B      for(i = 0; i<N; i++) {

            sum = sum + arr[i];

        }

        return sum;

}
```

Total memory allocated in bytes

⇒ 8 B

↓

O(1)

Space Complexity : ⇒ Amount of extra space taken by your algorithm other than Input space

K = 6

[1, 5, 3, 2, 6]

//

```
fun (int arr[], int N, int k) {

    for(i=0; i<N; i++){

        if(arr[i] == k) {

            return true;

        }

    }

    return false;

}
```

\# of iterations:

Best: 1 → $\boxed{\Omega(1)}$

Worst: N → O(N)

Try this advanced question

```
fun(int N, int k){
    for(i = 1; i <= N; i++){
        P = power(i, k);   // O(1)
        for(j = 1; j <= i^k; j++){
            [1,
            |
            |
        }
    }
}
```

5 min

# of iterations :-

i	j	iterations
1	$[1, 1^k]$ →	1^k +
2	$[1, 2^k]$ →	2^k +
3	$[1, 3^k]$ →	3^k +
⋮	⋮	⋮ +
N	$[1, N^k]$ →	N^k

$[1, i^k]$

No. of iterations :-

$$1^K + 2^K + 3^K + \cdots + N^K$$

$K=1$:- $[1 + 2 + 3 + 4 + \cdots + N]$ $\frac{N*(N+1)}{2} \Rightarrow \frac{N^2 + \cancel{N}}{2} \Rightarrow \frac{N^2}{2}$

$K=2$:- $[1^2 + 2^2 + 3^2 + \cdots + N^2]$ $\frac{N*(N+1)*(2N+1)}{6}$

Sum of squares of first N Natural Nos :-

$\Rightarrow \frac{2N^3}{6} \Rightarrow \frac{N^3}{3}$

$K=3$:- $[1^3 + 2^3 + 3^3 + \cdots + N^3]$ $\left(\frac{N(N+1)}{2}\right)^2$

$\Rightarrow \frac{(N^2)^2}{(2)^2} \Rightarrow \frac{N^4}{4}$

K :- $\left[\frac{N^{K+1}}{K+1}\right] \Rightarrow O\left(\frac{N^{K+1}}{K+1}\right)$

$O(N^K)$

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

N^K N

→ Arrays

Thank You :)

$\frac{N}{2}$

Doubts:-

```
for(i=1 to i<=N){
    if(i%2==0){
        for(j=1; j<=i; j++)
        {
        ____
}
```

$i = 2 \quad 2$

$i = 4 \quad 4$

$\boxed{N = 6}$

$i = 1 \rightarrow 1$

$i = 2 \rightarrow 2 \; [1, 2]$

$2 + 4 + 6$

$N = 8$

$i = 3 \rightarrow 1$

$2 + 4 + 6 + 8$

$i = 4 \rightarrow 4 \; [1, 4]$

$i = 5 \rightarrow 1$

6

Close Approximate

even Nos :$\rightarrow \frac{N}{2}$

$\frac{N}{2}$

a, r, N

i will even

$[2 + 4 + 6 + \cdots] \Rightarrow$

i will add

$\underbrace{[1 + 1 + 1 - - -]}_{\frac{N}{2}} \rightarrow N/2$

Ques fun(

```
sum = 0
for(i=0; i<N; i++){
    sum = sum + arr[i]
```

$\log_a N \Rightarrow$

log

$\boxed{\log_2 N} \Rightarrow$ The power to which 2 should be raised in order to get N

$\Downarrow$

Inverse exponents

$\boxed{\log_2 4} \Rightarrow 2$ $\boxed{N=4}$ $\boxed{2^2} = 4$

$\boxed{\log_2 16 \Rightarrow 4}$ $\boxed{N=16}$ $2^4 = 16$

$\log_2 3 \Rightarrow 1....$ $2^x = 3$

$$\log_3 9 \Rightarrow \log_3 3^2 \Rightarrow 2\log_3 3 \Rightarrow 2$$

Inverse exponent

$$\boxed{3^2 = 9}$$

$$\boxed{b^2 = N}$$

$\log_b N \Rightarrow$ The power to which b should be raised to get N.

$$\boxed{a^x} = \boxed{N} \Rightarrow \log_a a^x = \log_a N$$

$$\boxed{x = \log_a N} \Leftarrow x = \log_a N$$

Thank You