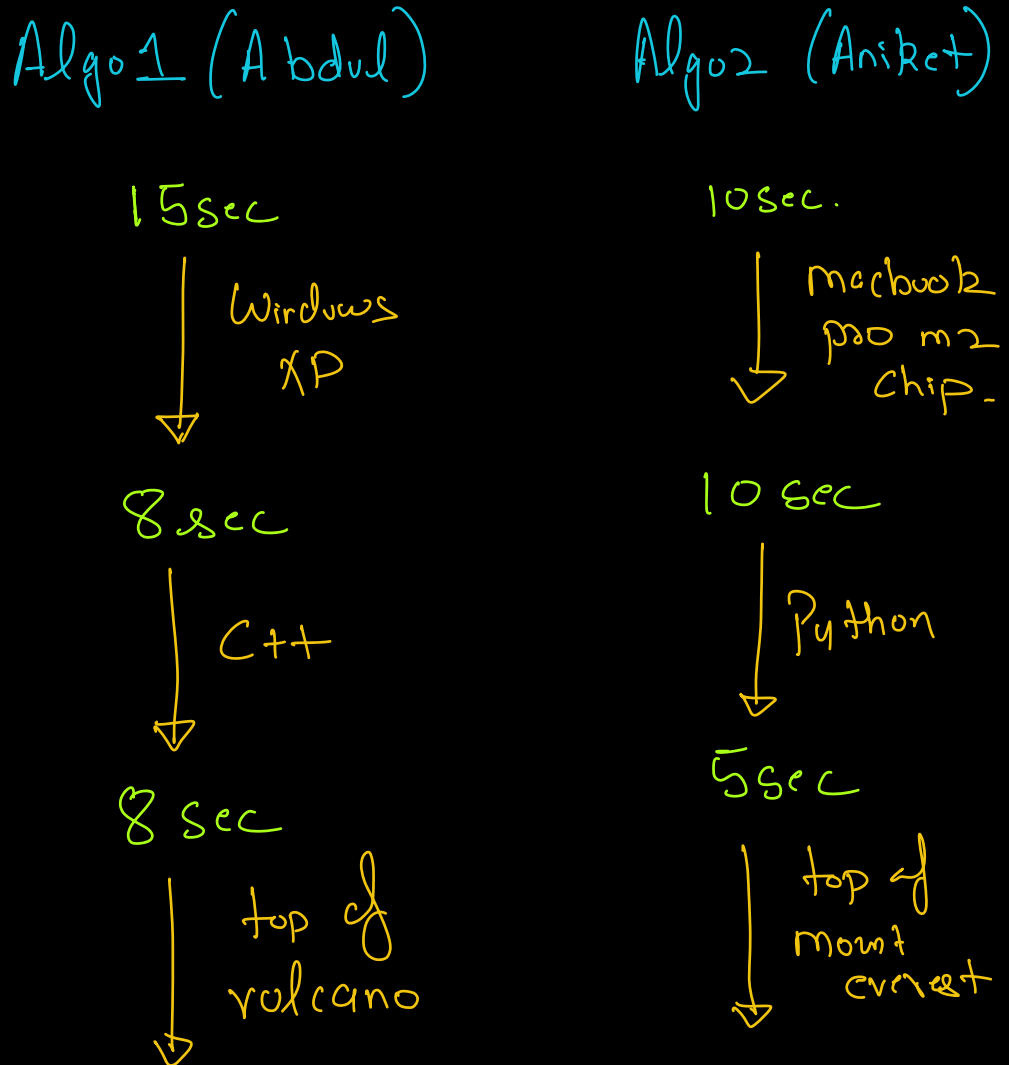
Q Given an array. Sort the numbers in decreasing order.

No of elements in array : 10^4.



CAN'T JUDGE

Execution time : Depends on external factors. So it is not right metric to compare 2 algorithms

→
```
for(int i=0 ; i<n ; i++) {
    print(i);
}
```

Iterations : n

⇒ It is better to compare algorithms by looking at the number of iterations.

Q Some random Question.

	Algo 1	Algo 2
Iterations :	$100 \log_2 n$	$n/10$

for $n < 13746$, $n/10$ is better

for $n \geq 13746$, $100 \log_2 n$ is better.

Real World :

Ind vs Pak : $\simeq$ 2cr, 20 million.

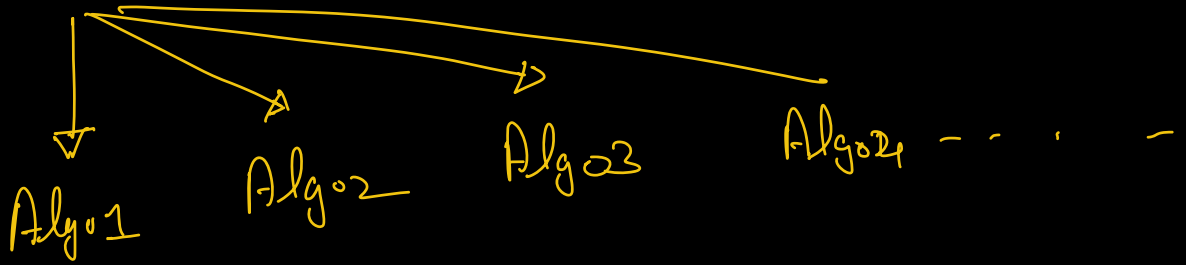
Google search results : $\simeq$ millions.

Youtube most video : Baby shark : 10 billion.

→ Choose algorithm which works better, for very large inputs.

Algo 1 is better

Q Some random Question.



Asymptotic Analysis of Algorithms

↳ Analysis of algorithms for very large input.

↳ tending to ∞.

Big O.

	Algo 1	Algo 2
Iterations :	$100 \log_2 n$	$n/10$
Big O :	$\log_2 n$	n.

Algo 1 is better than Algo according to Big O

To Calculate Big O

→ Independent of external factors.

1. → Calculate iterations based on Input Size
2. → Take higher order term & Neglect lower order terms
3. → Ignore Constants.

Neglect lower order terms

Iterations : $N^2 + 100N$

Input Size	Total iterations	Contribution by lower order term.
$N = 10$	$100 + 1000$	$\simeq 90\%$
$N = 100$	$10000 + 10000$	$\simeq 50\%$
$N = 10^5$	$10^{10} + 10^7$	$< 0.1\%$

As input size increases, contribution of lower term decreases.

For large inputs, the contribution is negligible.

Neglect Constants

Alog 1	Algo 2	Which is better for large inputs?
$10 \log N$	N	Algo 1
$100 \log N$	N	Algo 1
$10^3 \log N$	N	Algo 1
$10N$	$N^2/10$	Algo 1
$N \log N$	$100N$	Algo 2

$N < 2^{100}$ $N \log N$

$N > 2^{100}$ $100N$

Issues in Big O

Algo1 : $10^3 N$ Algo2 : N^2

Big O : $O(N)$ $O(N^2)$

CLAIM : For all input size, Algo 1 is better ✗

Input	Iterations (Algo1)	Iterations (Algo2)	Which is better.
$N=10$	10^4	10^2	Algo 2 is optimised.
$N=100$	10^5	10^4	Algo 2 is optimised.
$N=10^3$	10^6	10^6	Both are same.
$N=10^3+1$	$\{10^3\}\{10^3+1\}$	$\{10^3+1\}\{10^3+1\}$	Algo 1 is optimised.
$N=10^4$	10^7	10^8	Algo 1 is optimised.

For $n \leq 10^3$, Algo2 is better.

$n > 10^3$, Algo 1 is better

10^3

CORRECT CLAIM:

After Big O comparision Algo 1 is better than Algo 2 for all input value after a certain threshold.

Issues in Big O

Algo1 : $2N^2+4N$ Algo2 : $3N^2$

Big O : $O(N^2)$ $O(N^2)$

$2N^2+4N$ $2N^2+N^2$

→ Looking at the number of iterations, Algo1 is better.

Algo1 : $5N^3+6N^2$ Algo2 : $4N^3+10N$

Big O : $O(N^3)$ $O(N^3)$

→ Looking at the number of iterations, Algo 2 is better.

if Big O is same for 2 algorithms, the algo having lower coefficient for the highest order term is better.

Code → iterations → Big O

Problem : Search for an element = k.

```
bool search ( int k , int arr[] ) {
    int n = arr.length;
    for (int i=0 ; i<n ; i++) {
        if (arr[i] == k)
            return true;
    }
    return false;
}
```

iterations

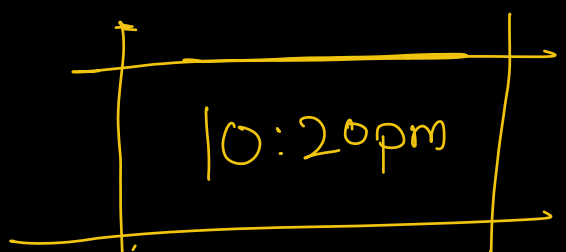
Best → 1

Worst → n.

Manager task.

5 days.

30 days.



Space Complexity

int ⇒ 4 bytes
long ⇒ 8 bytes.

1.
```
func (int N) {
    int x = N;
    int y = x^2;
    long z = 2x + y;
}
```

Total Space: 4+4+4+8
⇒ 20 bytes.

Space by algo: 4+4+8
⇒ 16 bytes.

SC: O(1)

2.
```
func (int N) {
    int x = N;
    int y = x^2;
    long z = 2x + y;
    int p[N];
}
```

Total Space
⇒ 4+4+4+8+4N

Space by algo: 4+4+8+4N
⇒ 16+4N.

SC: O(N)

3.
```
func (int N) {
    int x = N;
    int y = x^2;
    long z = 2x + y;
    int p[N];
    long mat[N][N];
}
```

Total Space $\Rightarrow 4+4+4+8+4N+8N^2$

Space by algo: $4+4+8+4N+8N^2$

SC: $O(N^2)$

Space Complexity : Extra space consumed by the algorithm to do the computations.

Auxiliary Space.

Computation Space

Q Find max of an array.

```
int max_value (int arr[]) {
    int ans = arr[0];
    for (int i = 1 ; i < n ; i++) {
        ans = max (ans, arr[i]);
    }
    return ans;
}
```

SC: O(1)

int ans_calc (int arr[]) {

```
    int pf[n];
    pf[0] = 0;
    for (int i = 1; i < n ; i++) {
        pf[i] = pf[i-1] + arr[i];
    }
}
```

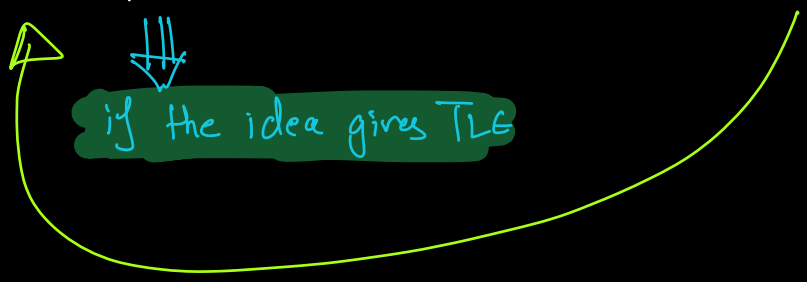
Space = 4N + 4

SC: O(n)

Time limit Exceeded [TLE]

X company. ⇒ Coding round 2 Question.

Q_1 : idea → code → Submit → TLE



Online Editors : Same processor ⇒ 1 GHz

⇓

perform 10^9 instructions per sec.

Your solution should get accepted in 1 sec.

⇒ At max our code can perform 10^9 instructions.

Pseudo Code

```
int count_factors (int N) {
    int c = 0 +1
    for (int i=1 ; i <= N ; i++) {        // +1, +1, +1
        if (N % i == 0) {                 // +1, +1
            c++;                          // +1
        }
    }
    return c
}
```

Total iteration = N.

Total instructions. ⇒ 6N or 7N.

Scenario 1 :

no of instructions per iteration : 10.

total instruction our code can perform : 10^9

Max iterations $\Rightarrow$ 10^8

$10^8 \times 10 \Rightarrow 10^9$

> Code can have at max 10^8

Scenario

per iteration $\Rightarrow$ 100 instruction.

Max iterations $\Rightarrow$ 10^7

> Code can have at max 10^7

no of iteration $\times 100 \leq 10^9$

no of iteration $\leq 10^7$

In general your code will be accepted if it has $[10^7, 10^8]$ iteration.

How to Solve a problem

- Read the Qus
- Looking at Constraints
- Think of idea.
- Correctness of idea
- Code.

Problem.

$1 \leq n \leq 100$

WORKS

$T_c: O(n^3)$

Problem.

$1 \leq n \leq 1000$

Max iterations $= 10^9$

Will not work

$T_c: O(n^3)$

Clues.

$1 \leq n \leq 10^3$

- $O(n^2) \not\Rightarrow 10^6$
- $O(n \log n) \leq 10^6$
- $O(n) \not\Rightarrow 10^3$
- $O(\log n) \leq 10^3$

Clues

$1 \le n \le 10^5$

- $O(n^2)$ ✗
- $O(n\sqrt{n}) \simeq [10^7, 10^8]$
- $n \log n$
- n

$O(n^2 m)$

$n \le 1000$

$m \le 10$

$10^6 \times 10 = 10^7$

int arr [10^4]

int arr [10^3] [10^3]

int arr [10^4] [10^4]

10^8

```
int n = 1000
    int i = 0
while (i < N) {
        i++;
}

int c = 0 , int i = 1;
while (true) {
    if (prime(i))
        c++;
    if (c == 100)
        break;
    i++;
}
```