Today's Content :-

- → Comparing 2 Algos
- → Why Big O needed
- → Space Complexity
- → TLE
- → TC naming conventions
- → Advance problem in calculating Iterations.

Contest :- Given n elements sort them in ascending order.

arr[5] = {3, 2, 6, 8, 13} → {1 2 3 6 8 3}

→ $N = 10^4$

Vishal (Algo1)

↓

15 sec

↓

(windows xp)

↓

(Macbook M2)

↓

7 sec

↓

C++

↓

(Top of hot volcano)

↓

Mt. Everest

↓

5 sec

Rohan (Algo2)

↓

10 sec

↓

(Macbook M2)

↓

10 sec

↓

Python

↓

C++

5 sec

↓

(Mt. Everest)

Execution time :- It depends on so many external factors, hence we generally don't compare, execution time b/w 2 algorithms.

iterations. `for (i=1; i<=N; i++) {` → `print(i)` → `}` (3)

iterations N.

Conl:- To compare 2 algorithms, calculate their iterations, based on input size and compare them.

Q) →

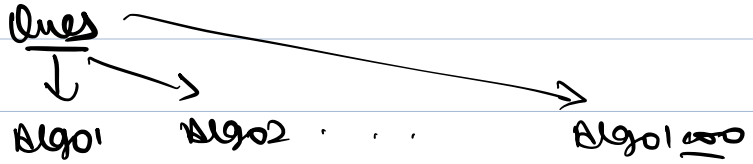
Algo 1 (Tanu)	Algo 2 (Saket)
$\log_2 N$	$N/10$

obs :-

Till	Iterations
$N <= 350$	Tanu > Saket
$N > 350$	Saket > Tanu

Ind us pak : 10 - 11 million google result : millions results.	Pick algorithm, which works better for very large input.

Col2:- →



Asymptotic Analysis of Algorithms

↳ Analyse performance of an algo for very large inputs.

↳ Big (O) → notation

Algo 1 (Tanv)

$\leq \log_2 N$

↓

$O(\log_2 N)$ $<$ $O(N)$

Algo 2 (Saket)

$N/10$

To calculate Big(O)

→ Calculate iterations

→ Take higher order terms (ignore lower order)

→ Ignore constant co-efficients.

Neglect lower order Terms ?

O → Algo (Priyanka)

Iterations N^2+10N

Input	Total Iterations	% of lower order contr. in total Iterations.
$N = 10$	200	$\frac{100}{200} \times 100 = 50\%$
$N = 100$	$10^4 + 10^3$	$\frac{10^3}{10^4+10^3} \times 100 \simeq 9.1\%$
$N = 10^4$	$10^8 + 10^5$	$\frac{10^5}{10^8+10^5} \times 100 \simeq 0.1\%$

Obs:- as Input size increases, contribution of lower order term decreases.

Neglect Constant Co-effecients

O1 →

Algo1 (Rowind)	Algo2 (Hima)	for large Inputs
$10 \log_2 N$	N	Rowind
$100 \log_2 N$	N	Rowind
$10^3 \log_2 N$	$N/10$	Rowind
$10\ N$	$\frac{N^2}{10}$	Rowind
$N \log N$	$100\ N$	Hima

//Issues in Big O

θ → Avinash (Algo1) Pawan (Algo2) more optimized.

$10^3 N$ N^2

Big(1) O(N) $O(N^2)$ Avinash always better?

Input

N = 10	→	10^4	10^2 → (Pawan)
N = 100	→	10^5	10^4 → (Pawan)
N = 10^3	→	10^6	10^6 → (both are same)
N = 10^3+1	→	$10^3(10^3+1)$	$(10^3+1)*(10^3+1)$ Avinash
N = 10^4	→	10^7	10^8 → Avinash

Claim :- for all Input ≥ 1000, Avinash is Better.

Final claim :-

when we compare 2 algo using Big O, Algo1 will be better than algo2, for all Input value above a certain point.

↳ Threshold point

1) After threshold Big(O) holds.
2) Don't worry about threshold value.

Issues in Big(O)

O →	Algo1 (Amrush)	Algo2 (chelam)
	$2N^2 + 4N$	$3N^2$
Big(O) →	$O(N^2)$	$O(N^2)$

(Both are same acc. to Big O)

Eg2:	$5N^3 + 6N^2$	$4N^3 + 10N$
Big(O)	$O(N^3)$	$O(N^3)$

(Both are same according to Big O)

Obs 2: If 2 algo's have same Big(O), we can't really compare with Big O, we need iterations to compare. (Co-efficients of higher order terms in iterations)

Code :- Searching for an element = x

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

T.C → O(n)

Iteration
- Best → 1
- Worst → n

Note :- while writing Big(O) we consider worst case iterations

Break :- 10:12 → 10:20 pm

$\left(\frac{3}{2}\right)^n$

Code → Time Complexity
Code → Space Complexity

Space Complexity

int → 4B
long → 8B.

```
void func(int N) {
    int n = N  +4
    int y = n*n  +4
    long z = n+y.  +8
}
```

Total memory = 20B.

S.C → Independent of Inp.
↳ O(1)

```
void func(int N) {
    int n = N   → 4
    int y = n^2  → 4
    long z = n+y  → 8
    // Declare an array.
    int[] arr = new int[N].
}
```

4N

Total memory = 4N + 20

S.C ⇒ O(N)

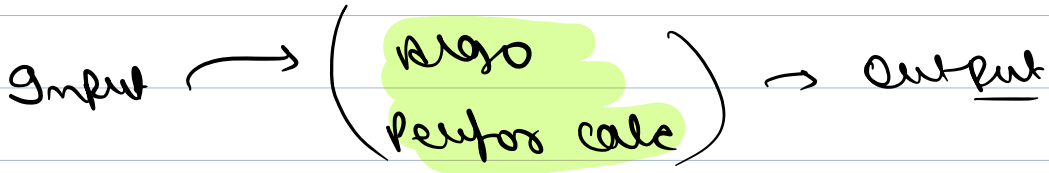
```
funct (int N) {
    int x = N        → 4
    int y = x^2      → 4
    long z = x+y.    → 8
    int [] ar = new int [N] : 4N

    long [][] l = new long [N] [N]
                  8N^2
}
```

S.C → $8N^2 + 4N + 20$

$\Rightarrow O(N^2)$

→ Algo we develop →



S.C. of an algorithm :- It is amt. of space additionally used by algorithm, other than input space to perform Necessary computation.

auxillary space

// Given a program to find max.

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

+4B, +4B

T.C → O(N)
S.C → O(1)

// Given an array todo some task.

```
int Task(int arr[]){
    int n = arr.length
    int pf[n]
    pf[0] = arr[0]
    for(i=1; i<n; i++){
        pf[i] = pf[i-1]+arr[i]
    }
    // perform something.
}
```

+4B, 4N, 4

T.C →

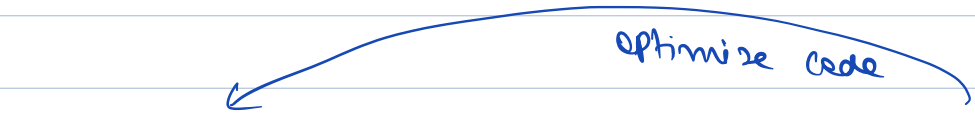
Iteration → n-1 → O(n)

S.C → (4N+8) → O(N)

TC > SC { In most cases you'll be asked to improve time first and then space }

TLE :- (Time Limit Exceeded)

↳ Mehta → (Amazon) → Hiring challenge → 2 Q → 1 hour.



Q1 → Idea → Code → Submit → TLE

→ without even writing a single line code, we can say whether logic will work or not.

Online Editors → code server

↳ Code time exceeds 1sec inc++

2sec Java.

4sec Python

↳ PS → 1 GHz

↓

10^9 instructions / sec.

Obs 1:- At most our code can have 10^9 instructions

↳ variable =
↳ operator
↳ function calling

Pseudo Code

```
bool countfactor(N) {
    int C=0          +1
    for(i=1; i<=N; i++) {      +1  +1  +1
        if (N%i==0)            +1  +1
            C=C+1              +1
    }                                      > per iteration 6 or 7 instructions
    return C         +1
}
```

Total instructions 6N or 7N.

App 1 :-

↳ In our code 1 iteration = 10 instructions

→ Our code can almost contains = 10^9 instructions

$= 10^8 * 10$ instructions

Our code can have at most 10^8 iterations

App 2 :-

In our code 1 iteration = 100 instructions

⇒ Our code can contain only 10^9 instructions

$= 10^7 \times 100$

we can have only 10^7 iterations.

In general → code iterations = $[10^7 - 10^8]$ iterations

Solve a Question

Read it
Understand it
Logic
Correctness of logic
Code

Constraints

Q :- Read the Question

Constraints

$1 <= N <= 10^5$

idea → Pseudocode
↓
T.C
↓
$O(N^2)$
↓
No need to code

you need → to optimize

Q Read it

Constraints.

$1 <= N <= 10^3$

idea → $O(N^2)$

yes

Q Read it

Constraints

$1 <= N <= 10^4$

idea → Pseudo → $O(N^2)$
↓
WA / MLE

→ Dream Companies :- (FAANGMN)

→ DSA → strong hold → Scaler → revise it → Amazon Hire

→ Resume

→ Tell me about yourself.

→ Core CS → OS → DBMS → System Design