Today's Content :-

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

God grant me the serenity,
to accept the things I can't change,
courage to change the things I can,
and wisdom to know the difference.

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

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

→ $N = 10^4$

Arpit (Algo 1)

↓

15 sec

↓

(windows xp)

↓

Macbook M2

↓

7 sec

↓

C++

↓

Top of Hot Volcano

↓

Mt Everest

↓

5 sec

Vishnu (Algo 2)

↓

10 sec

↓

(Macbook M2)

↓

10 sec

↓

Python

↓

C++

↓

5 sec

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

iterations N,

iterations:-

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

Concl:- To compare 2 algorithms, calculate their iteration based on input size & compare them

Ques)

Levin	Rakesh
Algo 1	Algo 2
$100 \log_2 N$	$N/10$

obs:-

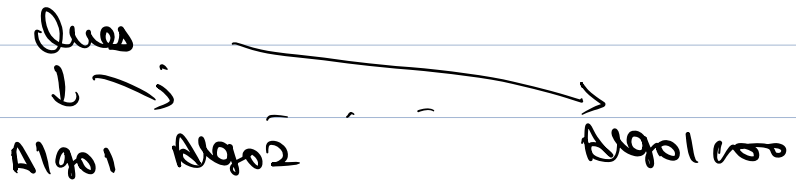
Till $N <= 3500$ after words — Levin > Rakesh

Rakesh > Levin

2nd - Pak 10-11 million

Google result > millions result

Pick algorithm which works better for very large input.



Asymptotic Analysis of Algorithms.

- Analyze performance of an algo for very large input.
- Big OH → Notation.

To calculate Big(O)

→ Calculate iterations

→ Take higher order terms (ignore lower order)

→ Ignore constant co-efficients.

$f(n) = 4N + 3N\log N + 10^6 \rightarrow O(N\log N)$

$f(n) = 4N\log N + 3N\sqrt{N} + 10^6 \rightarrow O(N\sqrt{N})$

Neglect lower order Terms ?

O → Algo (Ammel)

Iterations N^2+10N

Input	Total Iterations	% of lower order contr. in total Iterations.
$N=10$	200	$\frac{100}{200} \times 100 = 50\%$
$N=10^3$	10^6+10^3	$\frac{10^3}{10^6+10^3} \times 100 \simeq 9\%$
$N=10^4$	10^8+10^5	$\frac{10^5}{10^8+10^5} * 100 \simeq 0.1\%$

Obs:- As input size increases, contri. of lower order term decreases.

Neglect Constant Co-effecients

Q1 →

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

//Issues in Big(O)

	(Rohan)	(Saloni)	
	$10^3 N$	N^2	
Big(O)	O(N)	$O(N^2)$	Rohan always optimized ?.

Input

$N=10 \rightarrow$	10^4	10^2	Saloni
$N=100 \rightarrow$	10^5	10^4	Saloni
$N=10^3 \rightarrow$	10^6	10^6	Both are same
$N=10^3+1 \rightarrow$	$10^3(10^3+1)$	$(10^3+1)*(10^3+1)$	Rohan
$N=10^4 \rightarrow$	10^7	$10^8 \rightarrow$ Rohan.	

final claim:-

when we compare 2 algorithms using Big OH, Algo1 will be better than Algo2, for all input value above a certain point.

↳ Threshold point.

1) After threshold point Big OH holds.

2) Don't worry about threshold point.

Issues in Big (O)

	Algo1 (Yashwanth)	Algo2 (Prateek)
O →	$2N^2 + 4N$	$3N^2$
Big OH →	$O(N^2)$	$O(N^2)$

(Both are same acc to Big OH).

Eqn2:	$5N^3 + 6N^2$	$4N^3 + 10N$
Big (O)	$O(N^3)$	$O(N^3)$

(Both are same acc to Big OH).

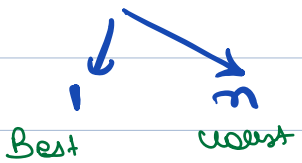
obs 2:- If 2 algos have same BigOH, we can't really compare with Big (OH), we need iterations to compare. (Co-effecient of higher order term)

Code :- Searching for an element = k

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

T.C → O(n)

Iterations :-



note:- while writing Big OH we consider worst case iterations.

Next class onwards → Code.

9:54 pm :- 10:04 pm. Break

Code → Time Complexity
Code → Space Complexity

Space Complexity

int → 4B
long → 8B.

```
— func (int n) {
|     int x = n
|     int y = n*n
|     long z = n+y.
3
```

Total memory:-
= 20B.

S.C → O(1)
↓
Space is independent of input.

```
— func (int n) {
|     int x = n
|     int y = n^2
|     long z = n+y
|     // Declare an array.
|     int [] arr = new int [n].
3
```

Total Memory = 20 + 4n

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
}
```

3

S.C → $8N^2 + 4N + 20$

$\Rightarrow O(N^2)$.

→ Algo we develop.

Input → (Algo perform calc) → output.

S.C of an algorithm:- It is amt of **space additionally** used by **algorithm** other than **input space** to perform **necessary computations.**

// Given a program to find max.

```
int maxarr(int arr[], int n) {
    int ans = arr[0];                    // 4
    for(i=1; i<n; i++) {                 // 4n
        ans = max(ans, arr[i])           // 3
    }                                    // 3
}
```

S.C → O(1)
T.C → O(N).

// Given an array todo some task.

```
int Task(int arr[]) {
    int n = arr.length                   // 4
    int pf[n]                            // 4n
    pf[0] = arr[0]
    for(i=1; i<n; i++) {                 // 4
        pf[i] = pf[i-1]+arr[i]           // 3
    }
    // perform something.
}                                        // 3
```

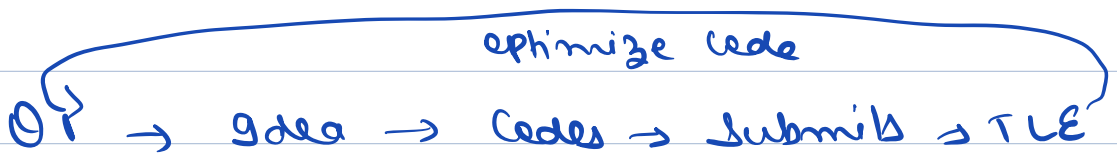
T.C → O(N)
S.C → 4N + 8 = O(N)

T.C > S.C {In most cases you'll be asked to improve time first & then space}

TLE :- (Time Limit Exceeded)

↳ Anuj → (Amazon) → Hiring challenge

↳ 20 → 1 hour.



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

Online Editors → code server

↳ code time

↳ Ps → 1 GHz.

↓

10^9 instructions/sec

C++ → 1 sec

Java → 2 sec

python → 4 sec

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

↳ variable =

↳ operators =

↳ fuction 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  +1
            c=c+1                +1
    }
    return c         +1
}
```

Per iteration 6 or 7 instruction

Total instructions 6N or 7N.

App1 :-

In our code 1 iteration = 10 instruction

our code can atmost contain

$= 10^9$ instructions.

$= 10^8 * 10$ instructions

our code can have at most 10^8 iterations.

App2 :-

In our code 1 iteration = 100 instructions

$\Rightarrow$ our code can contain 10^9 instructies

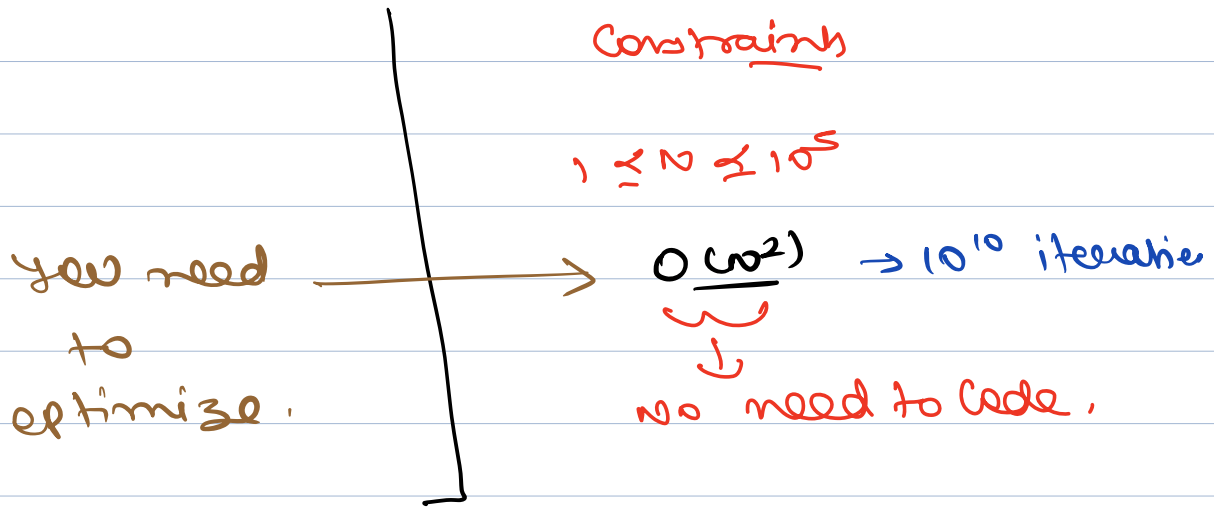
$= 10^7 * 100$

we can have only 10^7 iterations.

In general → Code iteration

$(10^7 - 10^6)$ iterations.

Solve a Question



Q Read it
Constraints
$1 <= N <= 10^3$
idea $\rightarrow O(N^2)$
yes.

Q Read it
constraints.
$1 <= N <= 10^4$
idea $\rightarrow O(N^2)$
↓
Wit / Min

MAANG

1) DSA
2) Resume
3) Tell me about yourself.
4) Computer fundamentals.
5) System Design ↓ OS, CN, DBMS.

Revision → After class
→ weekend
→ After month.