Todays Content:

a) Quizes on Big O ✓

b) Comparing 2 Algos ✓

   a) Using execution time ✓

   b) Using iterations & graphs ✓

c) Why Big O needed ? ✓

   a) Why only higher order terms considered ✓

   b) Why constant coefficient terms neglected ✓

   c) Issues in Big O ✓

      i) Working for large inputs ✓

      ii) Same Big O ✓

   d) Worst Case Iterations ✓

d) TLE : Time limit Exceeded

   a) Why TLE occurs & Working of Online Editors

   b) Importance of Constraints

Quizes on TC:

Big O:

a) Calculate Iterations ✓

Why Big O? ✓

b) Only higher Order Term ✓

c) Neglect Constant Coefficient ✓

	Higher Order Term	Big O
$f(n) = 3N^2 + 6N + 10^3 =$	$3N^2 \longrightarrow$	$O(N^2)$
$f(n) = 3N^2 + 6N^3 + 10^3 =$	$6N^3 \longrightarrow$	$O(N^3)$
$f(n) = 3N + 6N\log N + 10^3 =$	$6N\log N \longrightarrow$	$O(N\log N)$

Q) Given $N = 10^4$ elements sort them in increasing order?

ar[5] = { 3 2 6 8 1 } → { 1 2 3 6 8 }

→ 2 People Submitted their Algos: Both are executed on same Input

Execution Time : Time taken for system to run program

Algo1 : Chandini

(15sec) : WindowsXP

(8sec) Macbook M1

8 sec : C++ lang

8 sec : Very Hot

Algo2 : Suraj

(10sec) : Macbook M1

(10 sec) : Python

(6 sec) : C++

(6 sec) : Very Cool breezy

6sec Very Cool breezy (6 sec) : Very Cool breezy

Issues with Execution Time : It depends on lot of external factors

Hence not a correct way to compare 2 algorithms.

```
void func( int ar[] ){
    int n = ar.length
    int s = 0
    for(int i=0; i<n; i++){
        s=s+ar[i]
    }
}
```

a b = b - a + 1

Iterations : i : {0 .. N-1} = N-1-0+1 = N.

Iterations: It doesn't depend on external factors, hence we calculate & compare performance of 2 algos.

Comparing Iterations:

Q: Say for a Question following codes are submitted.

Nirpum (Algo1) Blue Deeksha (Algo2) green

Iterations: $100 \log_2 N$ $N/10$ Faster Code

$N < 3550$: Nirpum iterations < Deeksha iterations : Deeksha Code

$N > 3550$: Deeksha iterations > Nirpum iterations : Nirpum Code

In Real World: Data will keep increases

a) Ind vs Pak : 2cr

b) Google Search : million

c) Youtube : 100 cr

Obs: For larger inputs Nirpum better.

Same Comparison With Big O

Iterations:	$100 \log_2 N$	$N/10$	iterations	Code
Big O :	$O(\log_2 N)$	$O(N)$	N	$\log_2 N$

Assymptotic Analysis of Algorithms:

Analyzing performance of algorithms for very large inputs

3 Notations:

1. Big O 2. Omega 3. Theta : TODO Resources

How to Calculate Big O

1. Calculate no: of iterations
2. Consider only higher order Term & neglect Constant Coefficients

Why Consider only higher Order Terms?

Say Nerpum Code : $N^2 + 10N$ / Iterations lower order : $10N$

Input Size	Total Iterations	% lower order terms iterations in Total
$N = 10$	$f(10): 10^2 + 10 * 10 = 200$	$\frac{100}{200} * 100\% = 50\%$
$N = 100$	$f(100): 100^2 + 10 * 100 = 10^4 + 10^3$	$\frac{10^3}{10^4 + 10^3} * 100\% = \frac{10^5}{10^4 + 10^3} \approx 10\%$
$N = 10^4$	$f(10^4): [10^4]^2 + 10 * [10^4] = 10^8 + 10^5$	$\frac{10^5}{10^8 + 10^5} * 100\% = \frac{10^7}{10^8 + 10^5} \approx 0.1\%$

obs: As Input increases, contribution of lower order term less, hence neglect

Neglect Constant Coefficients : Comp happens for very large Inputs

	Algo1	Algo2	Iterations	Code faster
Q1 :	$10 \log_2 N$	N	N	$10 \log_2 N$
Q2	$100 \log_2 N$	N	→	$100 \log_2 N$
Q3	$10^3 \log_2 N$	$N/10$	→	$1000 \log_2 N$
Q4	$10N$	$N^2/10$	→	$10N$

obs2: Constant coefficients won't effect your comparisons for large Inputs

Issues in Bigo

	Algo1	Algo2		
	$10^3 N$	N^2	Iterations	Coac faster
Big O :	O(N)	O(N2)	Algo2	Algo1 is faster than Algo2 for all N ?*

Algo is faster than Algo2 for, after a certain input value / large value / ✓

N	Algo1 $10^3 N$	Algo2 N^2	Faster Algo is
10	$10^3 * 10 = 10^4$	10^2	Algo2 faster
10^2	$10^3 * 10^2 = 10^5$	10^4	Algo2 faster
10^3	$10^3 * 10^3 = 10^6$	$[10^3]^2 = 10^6$	Same
10^3+1	$10^3 * [10^3+1)$	$[10^3+1][10^3+1]$	Algo1 faster
10^4	$10^3 * 10^4 = 10^7$	$[10^4]^2 = 10^8$	Algo1 faster

If $N > 10^3$: Algo1 faster

Issues in Big O1:

	Algo1		Algo2	More iterations	Faster
	$10N^2$	vs	$2N^2 + 5N$	Algo1	Algo2
Big O :	O(N2)		O(N2)		

Note1: If we have same, Big O, then cannot compare.

Note2: If Big O same compare iterations

Imp: Code : Search for k in arr[]

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

Iterations: Best: ar[0]==k, 1 iteration

Worst: ar[0-1]==k or Not present, N iterations

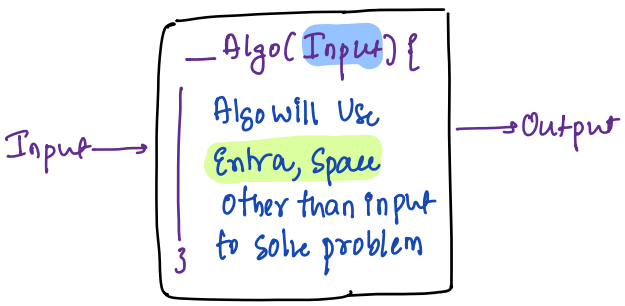
Note3: While calculating Big O we consider worst case iterations & calculate it.

Space Complexity

Code → Time Complexity : Calculate iterations & get Big O

Code → Space Complexity : Calculate Space Code takes & get Big O

Space Complexity Algo:



Note: When we calculate SC Space Complexity for code,
Don't take input memory for calculating SC for code.
Consider only Extra Memory taken by Algo.
Space algo Creates to Solve: Only that Considered

Examples:

```
                  ⟶ Input   Note: int: 4 B  long: 8 B
void func(int N){
|
|   Created by your Algo {
|       int x = n  ⟶ 4 B
|       int y = x*x ⟶ 4 B
|       long z = x+y ⟶ 8 B
|   }
3
```

Space = 16 Bytes ⟶ It is fixed / Constant

Big O ⟶ O(1)

```
void func(int N){                                   ← Input
    int x = N          ⟶ 4B
    int y = x^2        ⟶ 4B
    long z = x+y       ⟶ 8B
    int ar[] = new int[N] ⟶ int ar[] of N :
                             N int ele ⟶ 4N
}
```

Total Memory → 16B + 4N → Big O → O(N)

```
void func(int N){                                   ← Input
    int x = N          ⟶ 4B
    int y = x^2        ⟶ 4B
    long z = x+y       ⟶ 8B
    int ar[] = new int[N] ⟶ 4N
    int mat[][] = new int[N][N] ⟶ int mat[][] of N*N
                                   N*N int ele ⟶ 4N^2
}
```

Total memory → 16 + 4N + 4N^2 → Big O → O(N^2)

Q) Given a arr[N] write a program to get max of arr[]

```
int maxarr(int arr[]){                              ← given / Input Space
    int N = ar.length                  ⟶ 4B
    int ans = Integer.MIN_VALUE        ⟶ 4B
    for(int i=0; i<n; i++){            ⟶ 4B
        if(ans < ar[i]){
            ans = ar[i]
        }
    }
}
```

Total memory → 12B → Big O → O(1)

TLE : Time Limit Exceeded

Shukoor (Microsoft) → Hiring Challenge comin

Q1: Logic → Code → Submit → TLE : Time Limit Exceed

→ TLE

→ TLE

Idea: Without writing ___ logic can get TLE or not?

Working of Online Editors:

Code (Submit) —Run→ Online Servers —Process Speed→ 1 GHz = 10^9 Instruction/sec

In general Time limit in Online Server = 1 sec

Obs: Code can have at max 10^9 instructions

What 1 instruction
a. operator, +, −, /, %
<, >=, ==, =
b. function call
c. declaring variable

```
bool countFactrs(int N){
    int c = 0;          +1
    int i=1; i<=N; i++){
    +1        +1   +1
        +1  +1
        if(N%i==0){
            c=c+1;}     +2
    }
    return c;           +1
}
```

Iterations : i = [1..N] : N iterations

Instructions : 1+1+1+6N

Inside loop : 1 iteration = 6 instructions

Total instructions = 6N+3 ≈ 6N instructions.

For very code calculating iterations Tough.

Approach 1: Say in our Code 1 iteration = 10 instructions

↳ Code instructions limit = 10^9 instructions

$= 10^8 * 10$ instructions

$= 10^8 * 1$ iteration

} 10^8 iterations

Code Can have 10^8 iterations.

Approach 2: Say in our Code 1 iteration = 10^2 instructions

↳ Code instructions limit = 10^9 instructions $= 10^7 * 10^2$ instruction

$= 10^7 * 1$ iteration $= 10^7$ iterations

Code Can have 10^7 iterations

Con : Code Can have $10^7 \approx 10^8$ iterations

Question Structure

Question Desc: Given ar[N] ...do something

Constraints: $1 <= N <= 10^5$

$1 <= ar[i] <= 10^9$

Input Range:

$1 <= N <= 10^5$: min N = 1 max N = 10^5

$1 <= ar[i] <= 10^9$: min ar[i] = 1

max ar[i] = 10^9

Input format:

Output format:

Example Input:

Example Output:

Explanation:

Question:

a) Logic: Say Our Logic Nested loop?

TC: Based on Input Size : O(N2) // Idea → TC practice

Max $N = 10^5 \rightarrow O(N^2) \Rightarrow [10^5]^2 = 10^{10}$ iterations $> 10^8$ TLE

No need to Code

En1: Q) Given ar[N].. do something

Constraints:

$1 <= N <= 10^3$

$1 <= ar[i] <= 10^9$

Idea: → TC: O(N2)

Max $N = 10^3 \rightarrow O(N^2) = [10^3]^2 = [10^3 * 10^3] = 10^6$ $< 10^8$ work

Goto Code.

En2: Q) Given ar[N].. do something

Constraints:

$1 <= N <= 10^4$

$1 <= ar[i] <= 10^9$

Idea: → TC: O(N2)

Max $N = 10^4 \rightarrow O(N^2) = [10^4]^2 = [10^4 * 10^4] = 10^8$ on border

Touch ya so

Code ya check

Q: Given ar[N] calulate & return sum of ar[] ele

Constraints: $1 <= N <= 10^5$

$1 <= ar[i] <= 10^9$

MinSum:

ar[1] = {1}

mins = 1

MaxSum:

0	1	2	.....
10^9	10^9	10^9	.... 10^9

ar[10^5]

maxS = $10^9 * 10^5 = 10^{14}$

```
Sum(int ar[]) { TC: O(N)
    int s=0;
    long
    int N= ar.length;
    for(int i=0; i<N; i++){
        s=s+ar[i]
    }
    return s;
}
```

N = 10^5

$10^5 <= 2 <= 10^8$

Sum Range: 1 .. 10^{14} x Cannot store in int

int Range: $-2*10^9$ ... $2*10^9$

long Range: $-8*10^{18}$ ... $8*10^{18}$

Constraints

a) Will tell wether idea gives TLE or not

b) Datatypes we need to use for variables