We will start at 7:05 AM

Today's Agenda:-

1. Log Basics + Iteration Problems
2. Comparing Iterations using Graph
3. Time Complexity - Definition and Notations (Asymptotic Analysis - Big O)
4. TLE
5. Importance of Constraints

Log Basics

Q:- How to read the statement $\log_b a$?

:- To what power I should raise b so that I get a.

Examples

1. $\log_2 64$

$2^x = 64$

$x = 6$.

2. $\log_3 27$

$3^x = 27$

$3 * 3 * 3 = 27$; $x = 3$

3. $\log_5 25$

$5^2 = 25$

$x = 2$

4. $\log_2 32$

$2^x = 32$.

$x = 5$

$\log_2 10$

$2^x = 10$

$x = 3$ ....

$\log_2 40$

$2^5 = 32$

$2^6 = 64$.

$x = 5$.

Generalising,

$\log_2 N = K$.

$$2^K = N$$

$\log_2 2^6$; $2^x = 2^6$; $x = 6$.

$\log_3 3^5$; $3^x = 3^5$; $x = 5$.

Generalising. $\log_a a^N = N$

$\log_3 3^{10} = 10$

$\log_a a^1 = 1$

Q:- Given a positive integer N, how many times do we need to divide it by 2 until it reaches 1.

Ex:- N = 100

100 → 50 → 25 → 12 → 6 → 3 → 1

ans = 6.

Ex:- N = 324

324 → 162 → 81 → 40 → 20 → 10 → 5 → 2 → 1

ans = 8.

Quiz 1:-

9 → 4 → 2 → 1

ans = 3.

l to r → divide by 2.
r to l → multiply by 2

Generalisation :

$N \rightarrow N/2 \rightarrow N/4 \rightarrow N/8 \rightarrow \cdots 1$

$\frac{N}{2^0} \rightarrow \frac{N}{2^1} \rightarrow \frac{N}{2^2} \rightarrow \frac{N}{2^3} \rightarrow \cdots \frac{N}{2^k}$

$\frac{N}{2^k} = 1 \quad ; \quad 2^k = N$

$\log_2 N = k$

Quiz 2 :-

Divide 27.

$k = \log_2 27 \simeq 4..$

$27 \rightarrow 13 \rightarrow 6 \rightarrow 3 \rightarrow 1$

ans = 4

Quiz 3 :-

```
N > 0
i = N;
while (i > 1) {
    i = i/2;
}
```

$i \rightarrow i/2 \rightarrow i/4 \rightarrow i/8 \cdots 1$

$\log N$

$O(\log N)$

Quiz 4:-

```
for (i=1; i<N ; i = i*2) {
    .
    .
}
```

1 → 2 → 4 → 8 → 16 ⟶ ... N

$O(\log N)$.

Quiz 5

```
N >= 0
for (i=0; i<=N ; i=i*2)
{
    .
    .
}
```

i = 0000

Infinite.

Quiz 6

10 → `for ( i = 1 ; i <= 10 ; i++ ) {`

N → `for ( j = 1; j <= N; j++ ) {`

`. . . . .`

`}`

`}`

i	j	# iterations
1	[1, N]	N
2	[1, N]	N
3	[1, N]	N
⋮		⋮
10	[1, N]	N

$N * 10$

$= 10N$

$= O(N)$

Quiz 7 :

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

i	j	# iterations
1	[1, N]	N
2	[1, N]	N
3	[1, N]	N
⋮		
N	[1, N]	N

$N * N = N$.

$= O(N^2)$

Quiz 8 :

```
for (i = 1; i <= N; i++) {
    for (j = 1; j <= N ; j = j * 2) {
        3
    }
    3
}
```

i	j	# iterations
1	[1, N] *2	log N
2	[1, N] *2	log N
⋮		
N	[1, N] *2	log N

$O(N * \log(N))$.

Quiz 9:

```
for ( i = 1 ;  i <= 4 ;  i++ )  {
    for ( j = 1 ; j <= i ; j++ ) {
        3
    }
}
```

3

i	j	# iterations
1	[1, 1]	1
2	[1, 2]	2
3	[1, 3]	3
4	[1, 4]	4
5 ✗		

10.

= O(1)

Quiz 10:

```
for ( i = 1 ; i <= N ;  i++ )  {
    for ( j = 1 ; j <= i  ; j++ ) {
        :
        3
    }
}
```

3

i	j	# iterations
1	[1, N]	1
2	[1, 2]	2
3	[1, 3]	3
4	[1, 4]	4
:		
N	[1, N]	N

$1+2+3+4+\cdots N$

= Sum of first N natural nos.

$= \frac{N(N+1)}{2}$.

$= \frac{1}{2}N^2 + \frac{1}{2} \times N$ $= O(N^2)$

Quiz 11 :-

```
for (i = 1 ; i <= N ; i++) {
    for (j = 1 ; j <= (2^i) ; j++) {
        3
    }
}
3
```

i	j	# iterations
1	[1, 2]	2
2	[1, 2^2]	4
3	[1, 2^3]	8 $\rightarrow 2^3$
4	[1, 2^4]	2^4
:		
N	[1, 2^N]	2^N

$= a\frac{(r^N - 1)}{r-1}$

$= 2\frac{(2^N - 1)}{2-1}$

$2 + 2^2 + 2^3 + 2^4 + \cdots 2^N$

$a \quad ar \quad ar^2$

$= 2 \times 2^N - 2 = O(2^N)$

Comparing Iterations of Two Algorithms using Graphs.

Sumit's Algo

$100 * \log_2(N)$

Aravinth's Algo

$N/10$

Uptil N = 3500

Aravinth was better.

After 3500

Sumit's algo was better.

India Vs Pak. — 18 M.

Baby Shark → 2.8 B views.

We are always interested in larger inputs.

Asymptotic Analysis

Sumit's Algo → $100 * \log(N)$

Where:

→ $O(\log(N))$

Aravind's Algo → $\frac{1}{10} * N$

→ O(N)

Asymptotic Analysis or Big (O) simply means analysing algorithms for larger inputs

Steps to calculate Big (O):-

① Calculate Iterations based on Input Size.

② Ignore Lower Order Terms.

③ Ignore Constant Coefficients

Comparsion Order:

O(1)

log(N) < sqrt(N) < N < N log(N) < N sqrt(N) < N^2 < N^3 < 2^(N) < N! < N^N

Using an example

N = 36

$5 < 6 < 36 < 36*5 < 36*6 < 36^2 < 36^3 < 2^{36} < 36! < 36^{36}$

Examples :-

(1) $4N^2 + 3N + 1 = O(N^2)$

(2) $4N^2 + 3N + 6\,sqrt(N) + 9\log_2 N + 10$

$= O(N^2)$.

Quiz 12:

$4N + 3N\log(N) + 1$

$O(N \log N)$.

Quiz 13:

$4N\log N + 3N\,Sqrt(N) + 10^6$

$O(N\,Sqrt(N))$.

8:30.

Why do we neglect lower order terms ?.

Let total iterations $= N^2 + 10N$

Contribution of $10N$ (lower order Term)

$$= \frac{10N}{N^2 + 10N} \times 100$$

N	Total Iterations ($N^2 + 10N$)	Lower Order Term ($10N$)	% Contribution
10	200	100	50%
100	$10^4 + 10^3$	10^3	9 %
10^4	$10^8 + 10^5$	10^5	0.1 %

With increase in N, % contribution of lower order term reduces.

Why do we neglect constant Coefficients ?

Algo 1	Algo 2	Winner
Nikhil	Pooja	
$10 * \log N$	N	Nikhil.
$100 * \log N$	N	Nikhil.
$9 * N$	N^2	Nikhil
$150 * N$	$5 * N^2$	Nikhil.

Issues with Big(O):

Issue 1:- We can't say that one algo will always be better than the second algo.

Algo 1 $\longrightarrow$ $10^3 N$ $\longrightarrow$ $O(N)$

Algo 2 $\longrightarrow$ N^2 $\longrightarrow$ $O(N^2)$

Input Size	Algo 1	Algo 2	Winner
10	10^4	10^2	Algo 2.
100	10^5	10^4	Algo 2.
10^3	10^6	10^6	Both are same.
(10^3+1)	$10^3(10^3+1)$	$(10^3+1)(10^3+1)$	Algo 1.

Issue 2:- If 2 algos have same higher order terms Big O is not capable to identify better algorithm.

Ex:-

Print all the odd elements from 1 to N

Code 1:-

```
for (int i=1 ; i<=N; i++) {
    if (i-1.2 == 1) {
        print(i);
    }
}
```

N.

O(N)

Code 2:-

```
for (int i=1 ; i<=N; i+=2) {
    print(i);
}
```

N/2.

O(N)

Time Limit Exceeded (TLE) Error

Consider a student **Mehta** who likes **Amazon** and he is currently in a **hiring contest** organized by Amazon.

- He is given **two questions** and **1 hour** duration.
- He solves the **first** question, submits the code and finds it to be **TLE**.
- He again **changes the idea**, modifies the code and submits it, but he again gets **TLE**.

By the third time, the contest is already over.

- **Is it necessary to write the entire code and then test it to determine its correctness?**
- **Can we assess the logic's viability before writing any code?**

Yes.

Online Editors & Why TLE occurs ?

→ Code is executed on online servers of various platforms such as Codechef, Codeforces, etc.

→ The processing speed of their machines is 1 GHz which means they can perform 10^9 instructions per second.

→ Generally, codes should be executed in 1 sec.

Code

How many instructions ?.

```
bool countFactors(N) {                    +1
    int c = 0;                            +1
    for (i=1; i<=N; i++) {                +1  +1  +1
        if (N%i == 0) {                   +1  +1  +1
            c = c + 1;                    +1  +1
        }                                 3
    }                                     3
}                                         3
```

9. instructions in each iteration.

10 instructions in each iteration.

1 sec $\rightarrow 10^9$ instructions.

$\rightarrow 10^8$ iterations.

100 instructions in each iteration.

1 sec $\rightarrow 10^9$ instructions

$\rightarrow 10^7 * 10^2$ instructions.

1 iteration.

The total iteration my code can have are b/w $10^7 - 10^8$.

How to approach a problem?

- Read the **Question** and **Constraints** carefully.
- Formulate an **Idea** or **Logic**.
- Verify the **Correctness** of the Logic.
- Mentally develop a **Pseudocode** or rough **Idea of Loops**.
- Determine the **Time Complexity** based on the Pseudocode.
- Assess if the time complexity is feasible and won't result in **Time Limit Exceeded (TLE)** errors.
- **Re-evaluate** the **Idea/Logic** if the time constraints are not met; otherwise, proceed.
- **Code** the idea if it is deemed feasible.

Importance of Constraints ?

$$1 \leq N \leq 10^5$$

Complexity	Iterations	Works ?
$O(N^3)$	$(10^5)^3 = 10^{15}$	No.
$O(N^2 \log N)$	$10^{10} * \log 10^5$	No.
$O(N^2)$	10^{10}	No.
$O(N * \log N)$	$10^5 * \log 10^5$	Yes.

$$1 \leq N \leq 10^6$$

Complexity	Iterations	Works ?
$O(N^2 \log N)$	$(10^6)^2 * \log 10^6$	Never.
$O(N^2)$	$(10^6)^2 = 10^{12}$	Never.
$O(N * \log N)$	$10^6 * \log 10^6$	May work.
$O(N)$	10^6.	100% will work.

$1 \leq N \leq 100$, N^3 will also pass

$1 \leq N \leq 20$, 2^N will also pass.

$2^{20} = (2^{10})^2 = 1024 * 1024$

$\simeq 10^6$.

Next Class.

In next session we shall solve problems on Arrays. It'll be exciting to finally dive into the world of Problem Solving.

We'll solve every problem following certain set of steps:

✅Understanding the Problem Statement
✅Thinking about the Brute Force Approach
✅Making Observations for an Optimized Approach
✅Dry Running the Optimized Approach
✅Write the Code

Doubts

1. 3 6 12 24 48

$a = 3$

$r = 2$

$3 \quad 3\times2^1 \quad 3\times2^2 \quad 3\times2^3 \quad \dots$

$10^6 \quad 10^9 \quad 10^{12}$

2. $\log_{10} N$