TODAY:

1. Log Basics + Iteration problems.
2. Comparing iterations using Graph.
3. Time Complexity – Big O
4. T.LE
5. Importance of Constraints.

[ ] ⟶ Inclusive

( ) ⟶ Exclusive

$1 + 2 + 3 + 4 \cdots \cdots 100$ ⟶ A.P

$2 + 4 + 8 + 16 + 32 + 64$ ⟶ G.P.

$\log_{10} 1000 = \log_{10} 10^3 = 3$ //.

$\log_2 2^5 = 5$ //.

Log Basics

$\log_b a$ → To what value we need to raise b, such that we get a

$\log_b a = c$

$\log_{\underline{2}} \underline{64} = 2^6 = 6$ //.
(b, a)

$\log_3 27$: $b = 3$, $a = 27$,
$3^3 = 3$ //

$\log_5 25$: $b = 5$, $a = 25$
$5^2 = 2$ //.

$\log_2 32$: $b = 2$ $a = 32$
$2^5 = 5$ //.

$\log_4 16 = 2$

$\log_{10} 1000000 = 6$
10^6

$\log_2 (10) = 2^3$ (floor value)
$= 3$

$\log_2 (40) = 5$ //. (2^5)

$\log_2 2^{10} = 10$

If, $2^k = N$

$\log_2 2^k = \log_2 N$

$2^k = N$

$\log_2 2^k = \log_2 N$

$k = \log_2 N$

$k = \log_2 N$

$\log_2 9 = 2^3 = 3$ //.

100 → 50 → 25 → 12 → 6 → 3 → 1 (÷2 each step)

$\log_2 100 = 2^6$

$\log_2 2^6 = 6$ //

$\frac{2\times2\times2\times2\times2\times2\times2\times2\times2}{6}$

$\log_2 324 = 2^8 = 8$ //.

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

8 times

$9 \longrightarrow 9/2 \longrightarrow 9/4 \longrightarrow 1$ //

$N/2 \longrightarrow N/4 \longrightarrow N/8 \longrightarrow N/16 \cdots\cdots 1$

$N/2^1 \longrightarrow N/2^2 \longrightarrow N/2^3 \longrightarrow N/2^4 \cdots N/2^k$

$N/2^k = 1$

$N = 2^k$

$\log_2 N = \log_2 2^k$

$k = \log_2 N$

$k = \log_2 27 = \log_2 2^4 = 4$

$27/2 \xrightarrow{1} 13/2 \xrightarrow{2} 6/2 \xrightarrow{3} 3/2 \xrightarrow{4} 1$ ON=4

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

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

$N/2^1 \longrightarrow N/2^2 \longrightarrow N/2^3 \cdots N/2^k$.

$N/2^k = 1$

$N = 2^k$

$\log_2 N = \log_2 2^k$

$k = \log_2 N$

$9 \longrightarrow 9/2 \longrightarrow 9/4 \longrightarrow 9/8$

$9 \quad 4 \quad 2 \quad 1$

$9/2 \longrightarrow 4/2 \longrightarrow 2/2 \longrightarrow 1$

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

N = 32

$1 \xrightarrow{\times 2} 2 \xrightarrow{\times 2} 4 \xrightarrow{\times 2} 8 \xrightarrow{\times 2} 16 \xrightarrow{\times 2} 32$

$2^0 \longrightarrow 2^1 \longrightarrow 2^2 \longrightarrow 2^3 \longrightarrow 2^4 \longrightarrow 2^5$

$\log_2 2^5 = 5$ //

divide by 2 ⟶ 1 $(\log_2 N)$

1 and multiply by 2 ⟶ N $(\log_2 N)$.

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

i: 0 → 0 → 0 → 0 . . . . . ∞

```
for (i → 1 to 10) {        10*N
    for (j → 2 to N) {
        ;
    }
}
```

i	j	# iterations
1	1, 2, ... N	N
2	1 — N	N
3	1 - N	N
.		.
.		.
.		.
10	1 — N	N
		+
		10 N //

2

```
for (i ⟶ 1 to N) {        N * N = N^2
    for (j ⟶ 1 to N) {
        |
        3
}
```

i	j	iterations.
1	1 — N	N
2	1 — N	N
3	1 — N	N
⋮		⋮
N	1 — N	N

N * N.

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

N * log N

2

```
for (i=1; i<=4; i++) {  — 1→2
    for (j=1; j<=i; j++) {
        //
        3
}
```

i	j	iterations
1	1	1
2	1, 2	2
3	1, 2, 3	3
4	1, 2, 3, 4	4
5 (×)		10 //.

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

i	j	# iterations
1	1	1
2	1, 2.	2
3	1, 2, 3	3
4	1, 2, 3, 4	4
⋮		
N	1 —— N	N

AP = $1+2+3+4+5+\ldots\ldots N$

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

```
for (i=1; i<=N; i++) {          --> 1
    for (j=1; j<=2^i; j++) {
        // Print (i+j)
    }
}
```

i	j	# iterations
1	[1 - 2]	2
2	[1 - 4]	4
3	[1 - 8]	8
⋮		
N	[1 - 2^N]	2^N

10:20 ↓ 10:30 pm

GP series ⟶ $2+4+8+16+32+64\ldots 2^N$?

$r=2 \quad a=2, \quad T=N$

$= \frac{a\times(r^T-1)}{r-1} \qquad \frac{2(2^N-1)}{2-1} = \boxed{2(2^N-1)} = 2\times 2^N - 2$

Comparing iterations :-

Q: Say for a quit^n following codes are submitted.

Algo1: Atchya (higher value). Algo2: Sathya (factor)

iterations	$100 \log_2 N$	$N/10$	
$N < 3500$	sathya & Atchya	.	factor -> sathya
$N > 3500$	atchya & sathya	.	factor -> Atchya.

In real world :-

a) RCB vs CSK: 29.9 cr

b) Google search: millions

c) youtube: baby shark... 13.93 billion.

Some Comparision with Big(O)

	iterations	$100 \log_2 N$ (1)	$N/10$ (2)	iterations	code factor
Big O :		$O(\log_2 N)$	$O(N)$	Algo1	Algo1 ✓

Assymptotic Analysis of Algorithms.

3 Notations

1. Big O 2. Omega 3. Theta.

How to calculate Big O.

1. Calculate iterations
2. Ignore lower order terms (Consider Higher order terms).
3. Ignore Constant co-efficients

$10 N^2 + 4N \longrightarrow N^2 + N \rightarrow O(N^2)$

(10 → constant co-efficient; 4 → lower order)

$3N^2 + 1000N + 300 \longrightarrow N^2 + N \rightarrow O(N^2)$.

$\frac{N^3}{2} + \frac{N^2}{3} + \frac{N}{2} + 1 \rightarrow O(N^3)$.

$2^N + N^3 \longrightarrow O(2^N)$

Comparision order

$\log N < \sqrt{N} < N < N\log N < N\sqrt{N} < N^2 < N^3 < 2^N < N! < N^N$

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

Ques $4N\log N + 3N\sqrt{N} + 10^6 \rightarrow O(N\sqrt{N})$.

Why Consider Higher order terms and neglect lower order terms

Say code : $N^2 + 10N$ / iterations lower order: $10N$

Input size	Total iterations	% lower order terms iterat^n in total
$N = 10$	$10^2 + 10 \times 10 = 200$	$\frac{100}{200} \times 100 = 50\%$ → 50%.
$N = 100$	$100^2 + 10 \times 100 =$	$\frac{10^3}{10^4 + 10^3} \times 100 = 10\%$ → 90%.
$N = 10^4$	$(10^4)^2 + 10 \times 10^4 =$	$\frac{10^5}{10^8 + 10^5} \times 100 = 0.1\%$ → 99.9%.

obs: As i/p Increases, Contribut^n of lower order term decreases. Negleg.ble at 0.1%.

Neglect Constant Co-efficients?

	Ashita Algo1	Abhishek Algo2	move iterations	code faster ?
Q1:	$10 \log_2 N$	N	N	Ashita
Q2:	$100 \log_2 N$	N	N	Ashita
Q3:	$10^3 \log_2 N$	$N/10$	$N/10$	Ashita
Q4:	$10N$	$N^2/1000$	$N^2/1000$	Ashita

obs2: Constant co-efficients wont effect your comparision for very large inputs.

Issues in Big O :-

	Algo 1	Algo 2	
	$10^3 N$	N^2	iterations
Big O :	$O(N)$	$O(N^2)$	

code faster.

Algo 1 : for all values of N

Algo 1 will be faster ... ?

very large input value.

N	Algo 1 $10^3 N$	Algo 2 N^2	faster Algo is
10			
10^2			
10^3			
10^3+1			
10^4			

Issues in Big O

Algo 1		Algo 2	more iterations	faster
$10N^2$	vs	$2N^2+5N$		

Big (O)

Note 1 :

Note 2 :

Imp code :

```
func search (int arr[], int k) {
    for( i=0; i<N; i++){
        if (arr[i] == k) { return true; }
    }
}
```

Best : arr[0] == k — 1, 1 iter

Worst : arr[n-1] == k / k — N iterations

Note 3 :-

TLE: (Time limit exceeded).

(Google) → Hiring challange. → 2 questions 45 min.

Q1:

working of Online Editors

Code (submit) —Run→ Online servers —Process speed→

In general time limit in online servers = 1 sec

obs: Code can have at max:

what 1 instruction?

a. +, -, *, /, %.
   <, =, >
b. function call
c. declaring variable

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

iterations:

instructions:

Inside loop: 1 iteration:

total instructions =

for every code calculating ikr's

Approx: Say in one code 1 ikrations =

Code limit ikr's =

=

=

Code can have max 10^8 iterations

Approx: Sy in one code we have =

=

Question Structure:

1) Question Desc
   Constraints —
2) I/p format
3) o/p format
4) Example i/p
5) Example o/p
6) Explanation.

Importance of Constraint:

The next session shall start with Introduction to Space Complexity.

Then we shall dive into 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

I can sense your excitement, and I'm thrilled too!

MAKE SURE YOU ALL SOLVE TODAY'S ASSIGNMENT PROBLEMS and let me know in the WA group.