Agenda

1) Log Basics + Iteration Problems
2) Compare 2 Algo using graph
3) Time Complexity
4) TLE
5) Importance of Constraints

Basics of Log

⇒ Inverse of exponential function

$\log_b a$ ⇒ Log a to the base b.

⇒ What power do we raise b so that it becomes equal to a

$$\log_b a = c$$

$$\Rightarrow b^c = a$$

Eg

1) $\log_2 64 \Rightarrow 6$

$\Rightarrow 2^6 = 64$

2) $\log_3 27 \Rightarrow 3$

$\Rightarrow 3^3 = 27$

3) $\log_5 25 \Rightarrow 2$

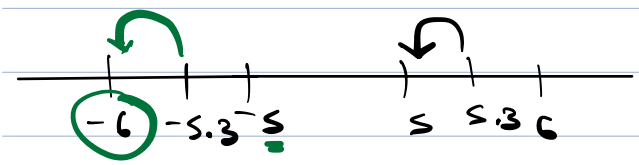
4) $\log_2 32 \Rightarrow 5$

$\Rightarrow$ Calculate the floor values of log

5.3

5. ~~—~~

↓ floor



5) $\log_2 10 \Rightarrow$ floor $(3.\_)$ $= 3$

$2^3 = 8$

$2^{3.\_} = 10$

$2^4 = 16$

6) $\log_2 40 \Rightarrow$ 5

$2^5 = 32$

$2^{5.-} = 40$

$2^6 = 64$

If $N = 2^K$

$\log_2 N = \log_2 2^K = K$

$2 \Rightarrow 2^{(6)}$ Ans

Given a positive no. N.
How many times do we divide it by 2 (consider the integer part while dividing) so that it becomes 1.

Eg: 100 → 50 → 25 → 12 → 6 → 3 → 1

⇒ 6 times.

Sol^n floor ($\log_2 N$)

$N \xrightarrow{1} \frac{N}{2} \xrightarrow{2} \frac{N}{4} = \frac{N}{2^2} \xrightarrow{3} \frac{N}{2^3} \cdots\cdots\rightarrow \frac{N}{2^K}$ (K times)

$$\frac{N}{2^K} = 1$$

$$N = 2^K$$

Take $\log_2$ both sides

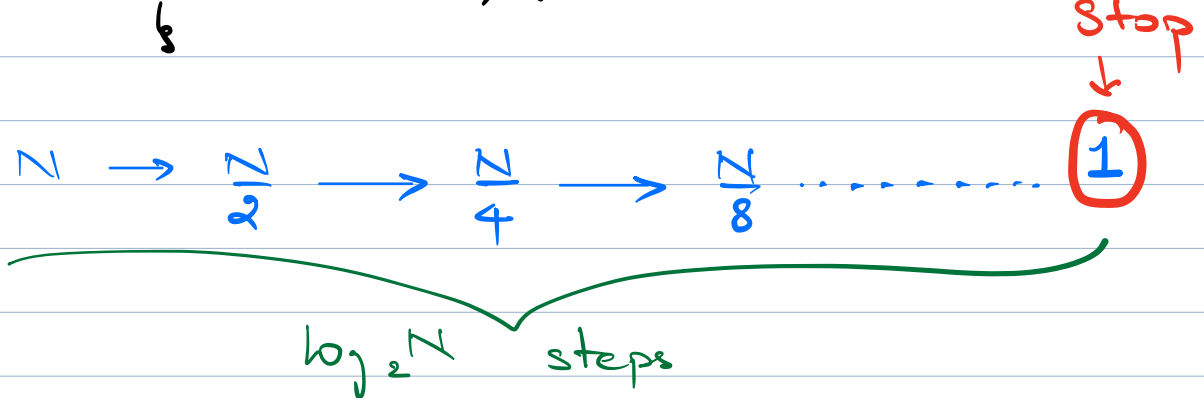
$$\log_2 N = \log_2 2^K \quad (K)$$

$$K \text{ (steps)} = \log_2 N$$

Iterations cont.

1)

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



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

`.....`

`}`

$1 \rightarrow 2 \rightarrow 4 \rightarrow 8 \ldots\ldots\ldots 2^{K}$ ⇒ (K) steps

$2^K = N$

$K = \log_2 N$

2) $N \geq 0$

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

`.....`

`}`

$0 \xrightarrow{\times 2} 0 \xrightarrow{\times 2} 0 \xrightarrow{\times 2} 0 \ldots$ (Infinite loop)

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

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

`.....`

10 × N

{

{

i	j	Iterations
1	[1, N]	N
2	[1, N]	N
3	.	.
4	.	.
5	.	.
6	.	.
7	.	.
8	.	.
9	.	.
10	[1, N]	N
		10 x N

5) `for (i = 1; i <= N; i++) {` — N x N

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

. . . . . .

}

}

6)

```
for (i = 1;  i <= N;  i++) {                    // N
    for (j = 1;  j <= N;  j = j * 2) {          // log2N
        ..... .....
    }
}
```

$N \times \log_2 N$

---

7)

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

$[1, i] = i - 1 + 1 = i$

i	j	Iterations
1	[1, 1]	1
2	[1, 2]	2
3	[1, 3]	3
4	[1, 4]	4
		10

8)

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

i	j	Iter.
1	[1, 1]	1
2	[1, 2]	2
3	[1, 3]	3
⋮		⋮
N	[1, N]	N

Sum of 1st N natural no.

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

---

9)

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

i	j	Iter
1	$[1, 2^1]$	2^1
2	$[1, 2^2]$	2^2
3	$[1, 2^3]$	2^3
4	$[1, 2^4]$	2^4
⋮		⋮
N	$[1, 2^N]$	2^N

$$2^1 + 2^2 + 2^3 \ldots\ldots + 2^N$$

N

$a = 2$

$r = 2$

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

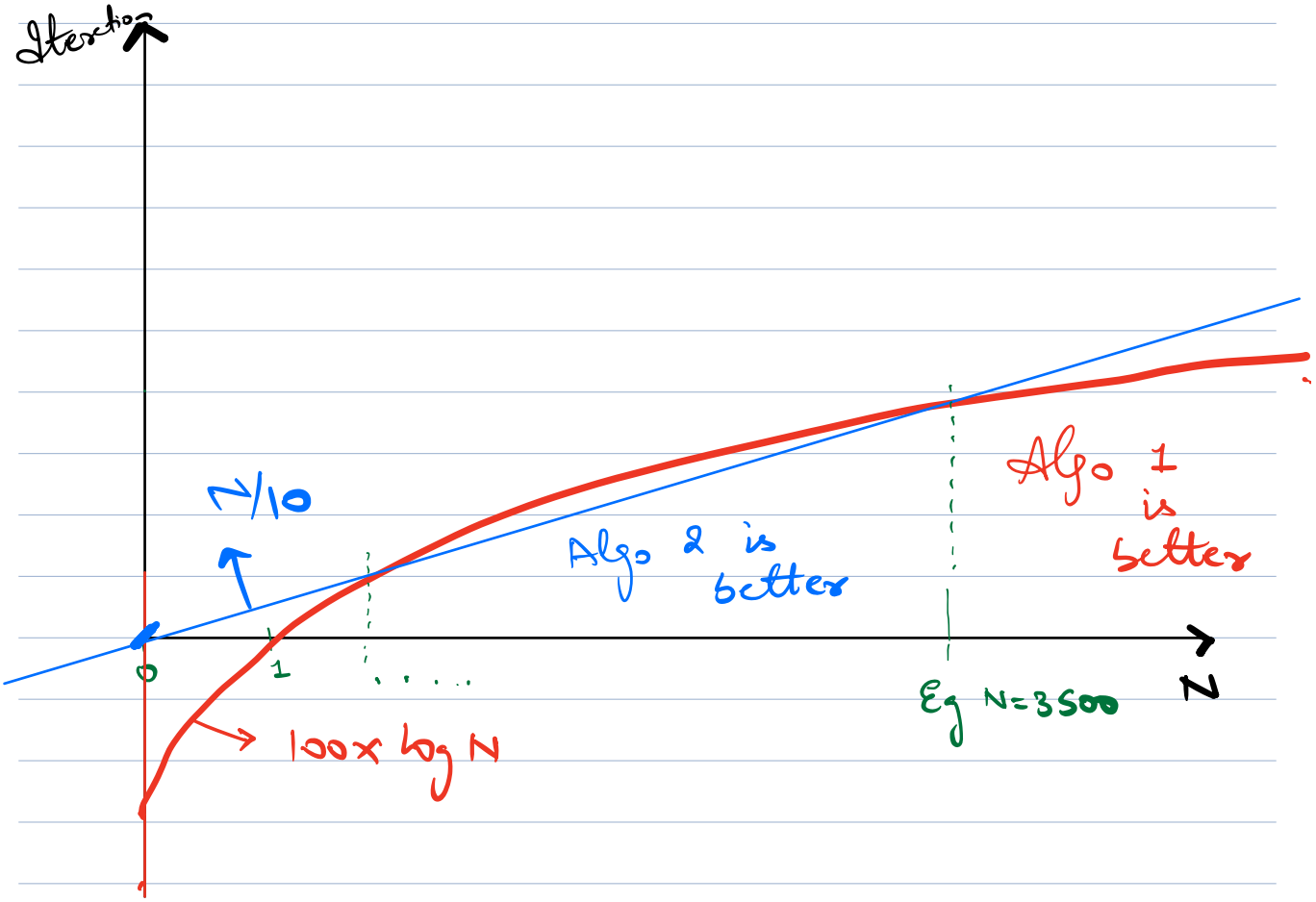
(r − 1) → 1

Compare 2 Algo using Graphs

Algo 1 : $100 \times \log_2(N)$

Algo 2 : $N/10$

Graph of these algo wrt N.



Scale of Real World Systems

Ind v/s Pak → 20M+

Baby Shark → 9.8B views.

Analysis of Algorithms ⇒ Huge Input Size

⇒ Asymptotic Analysis of Time Complexity

Big (O)

Steps to Calculate Big O Time Complexity

1) Number of iterations.
2) Ignore lower order terms.
3) Ignore the co-efficients of highest order term.

Eg: Iterations = ~~4~~ N^3 + ~~3~~ N^2 + ~~2~~ N + ~~5~~

T.C. = $O(N^3)$

Comparison Order

$$C < \log N < \text{sqrt}(N) < N < N\log N < N\,\text{sqrt}(N) < N^2 < N^2\log N < N^3 < 2^N < N! < N^N$$

$N = 36$

$$5 < 6 < 36 < 36\times 5 < 36\times 6 < 36^2 < 36^2\times 5 < 36^3 < 2^{36} < 36! < 36^{36}$$

Every question $\Rightarrow$ Big O

Time Complexity

Rate of growth of execution time with increase in I/P size.

$f(N) = \cancel{4N} + \cancel{8}N\log(N) + \cancel{2}$

$O(f(N)) = N\log N.$

$f[N] = 4N\log N + 8N\,\text{sqrt}(N) + 10^6$

$O(f[N]) = N \times \text{sqrt}(N)$

Why do we neglect lower Order Terms. ??

Iterations $= N^2 + 10N$

N	Total Iterations	Lower Order Iterations	Percentage Contri
10	$100 + 10\times10 = 200$	$10\times10 = 100$	$\frac{100}{200} \times 100 = 50\%$
100	$10^4 + 10^3$	10^3	$= \frac{10^3}{10^4 + 10^3} \simeq 9\%$
10^4	$10^8 + 10^5$	10^5	$\frac{10^5}{10^8 + 10^5} \times 100 \simeq 0.1\%$

With increase in I/P size, contribution of lower order term decreases & eventually become negligible.

Algo 1	Algo 2	Winner
$10 * \log_2 N$	N	Algo 1
$N = 2^{32}$	2^{32}	
$10 \times 32 = 320$	$\approx 10^9$	
$100 * \log_2 N$	N	
$N = 2^{32}$	2^{32}	Algo 1
$100 \times 32 = 3200$	$\approx 10^9$	

Issues with Big O Notations

Issue 1 :

Incorrect comparison for smaller inputs

Issue 2 :

If 2 algorithms have same Big O,

then we treat them as equals.

Algo 1	Algo 2
$3N^2$	$5N^2$
$\|\|$	$\|\|$
$O(N^2)$	$O(N^2)$

Time limit Exceed.

Suyash ⇒ Online contest test of amazon

1 hour duration
2 questions

Q1 ⇒ Submits ⇒ TLE

⇓

Changes idea Submits ⇒ TLE

⇓

Changes

⇓

Time Over

Can we define whether a solution will give TLE w/o submitting ??

Code ⇒ Online servers of Codechef, GFG, Hackerearth

Processing speed of server → 1 GHz

||

10^9 instructions per second

Generally Time limit for code = 1 sec

Our code can have at max 10^9 instruct.

Code

```
int countfactors (N) {
    int c = 0                          +1
    for (i = 1;  i <= N;  i++) {       +1   +1   +2 (i = i+1)
        if (N % i == 0)                +1   +1   } // N iterations
            c++;                       +2   }  8 instructions
    }
}
```

return c + 1;

}

Small Code ⇒ Approx 10 instruct.

Big Code ⇒ Approx 100 instruct.

[10, 100] ⇒ 10^9 instructions.

10 instruc. ⇒ $\frac{10^9}{10} = 10^8$ iterations

100 instruc. ⇒ $\frac{10^9}{100} = 10^7$ iterations

Approx, code ⇒ $[10^7, 10^8]$ iter.

General Steps to solve problem

1) Read Question & Constraints carefully
2) formulate Idea or Logic
3) Verify the correctness of logic

4) Mentally visualize the iterations.

5) Define T.C.

6) Assess whether this T.C. will give TLE or not (Using constraints)

Constraints

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

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
$O(N \log N)$	$= 10^5 \times \log 10^5$	Yes
	$\approx 10^5 \times 20$	