SCALER

Time Complexity

TABLE OF CONTENTS

.

Log Basics

⇒ inverse of exponential function.

$\log_b a$ → To what power b must be raised such that it becomes equal to a.

1. $\log_2 64 = 6$. $2^? = 64$
2. $\log_3 27 = 3$ $3^? = 27$
3. $\log_2 32 = 5$ $2^? = 32$
4. $\log_2 10 = 3$ (floor) $2^? = 10$ $2^3 \to 8$, $2^4 \to 16$
5. $\log_2 40 = 5$ (floor) $2^? = 40$ $2^5 \to 32$, $2^6 \to 64$
6. $\log_2 2^6 = 6$
7. $\log_3 3^5 = 5$

$[\log_a a^N = N]$

$2^K = N$

$\log_2 2^K = \log_2 N$

$K = \log_2 N$

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

< *Question* > : Given a positive integer N. How many times do we need to divide it by 2 until it reaches 1?

⇨ $\log_2 N$

N = 100

100 ↓/2 50 ↓/2 25 ↓/2 12 ↓/2 6 ↓/2 3 ↓/2 1

ans = 6

N = 324

324 ↓/2 162 ↓/2 81 ↓/2 40 ↓/2 20 ↓/2 10 ↓/2 5 ↓/2 2 ↓/2 1

ans = 8

N = 9

9 ↓/2 4 ↓/2 2 ↓/2 1

ans = 3

N = 27

ans = 4

$N \rightarrow \frac{N}{2} \rightarrow \frac{N}{4} \rightarrow \frac{N}{8} \rightarrow \frac{N}{16} \rightarrow \cdots \rightarrow 1$

$N \rightarrow \frac{N}{2^1} \rightarrow \frac{N}{2^2} \rightarrow \frac{N}{2^3} \rightarrow \frac{N}{2^4} \rightarrow \cdots \rightarrow \frac{N}{2^k}$

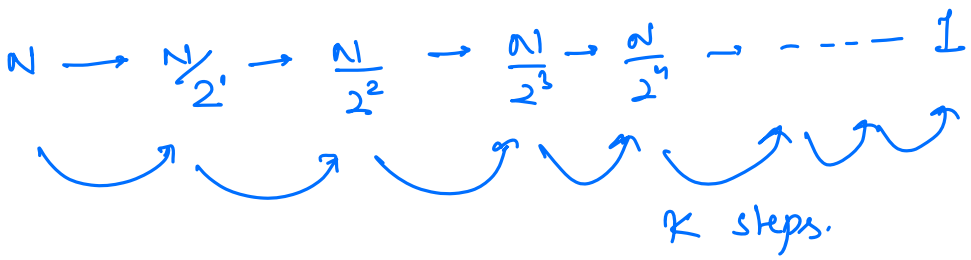
$\frac{N}{2^k} = 1 \Rightarrow N = 2^k \Rightarrow \log_2 N = \log_2 2^k$

$\boxed{\log_2 N = k}$

Quiz- 1

N>0

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



after K steps, loop will stop.

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

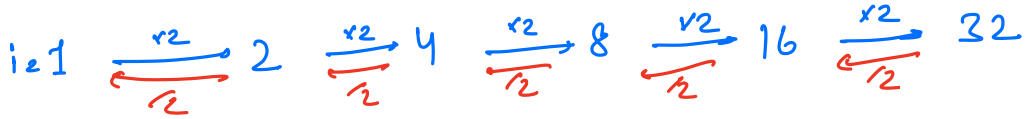
$[\log_2 N = K]$

$\therefore$ iterations $= \log_2 N$

Quiz- 2

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

$\therefore$ Iterations $= \log_2 N$



$i=1 \rightarrow 2^1 \rightarrow 2^2 \rightarrow 2^3 \rightarrow 2^4 \rightarrow 2^5 \text{ --- } 2^K$

$2^K = N \Rightarrow [K = \log_2 N]$

Quiz- 3

$N \geq 0$

```
for(i=0; i≤N; i=i*2){
    ------------
}
```

$i = 0 \xrightarrow{\times 2} i = 0 \xrightarrow{\times 2} i = 0 \xrightarrow{\times 2} i = 0 \xrightarrow{\times 2} i = 0 \xrightarrow{\times 2} i = 0$

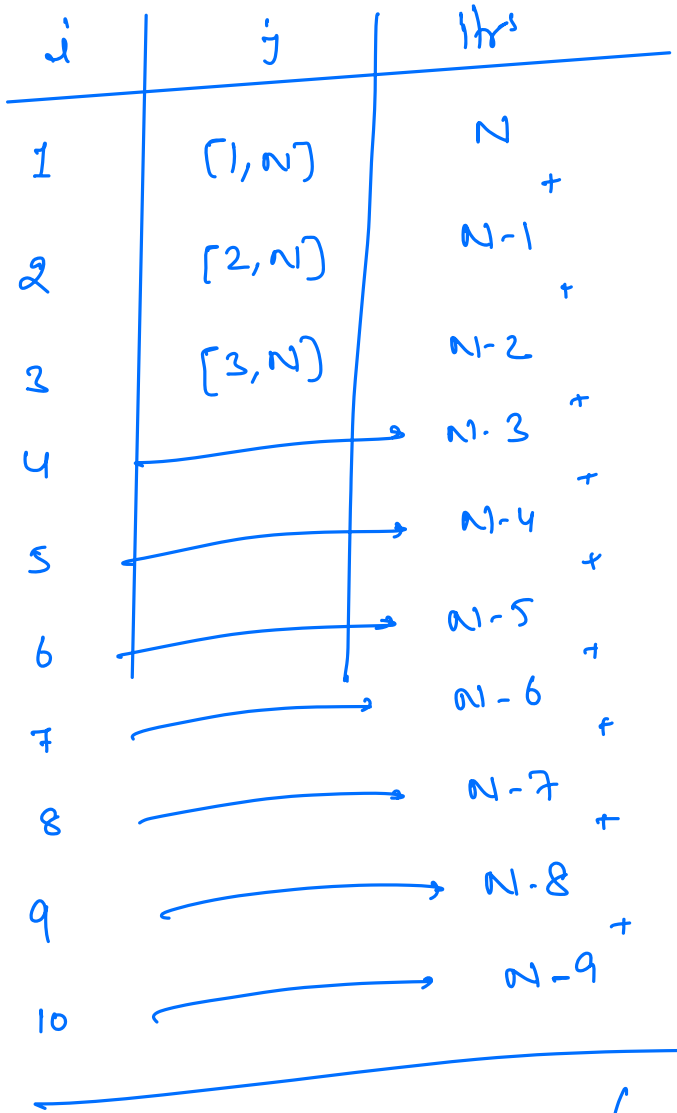
→ infinite iterations.

Quiz- 4

```
for(i=1; i≤10; i++){
    for(j=1; j≤N; j++){
        ---------------
    }
}
```

i	j	iterations
1	[1, N]	N +
2	[1, N]	N +
3	[1, N]	N +
1		1 +
1		1 +
10	[1, N]	N
		10 · N

```
for( i = 1; i <= 10; i++){
    for( j = i; j <= N; j++){
        ____
    }
}
```



$$10.N - \left(1+2+3+\cdots 9\right)$$

$$10N - \left(\frac{9\times 10}{2}\right)$$

$$\Rightarrow \underline{10N - 45}.$$

Quiz- 5

```
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
1	1	1
1	1	1
N	[1, N]	N

$N \times N = N^2$

Quiz- 6

```
for(i=1; i≤N; i++){
    for(j=1; j≤N; j=j*2){
        ----------
    }
}
```

i	j	iterations
1		$\log_2 N$
2		$\log_2 N$
3		$\log_2 N$
1		
1		
N		$\log_2 N$

$N \times \log_2 N$

Quiz- 7

```
for(i=1; i≤4; i++){
    for(j=1; j≤i; j++){
        //print(i+j)
    }
}
```

i	j	iterations
1	[1,1]	1
2	[1,2]	2
3	[1,3]	3
4	[1,4]	4

10 iterations.

Quiz- 8

```
for(i=1; i≤N; i++){
    for(j=1; j≤i; j++){
        //print(i+j)
    }
}
```

i	j	iterations
1	[1,1]	1
2	[1,2]	2
3	[1,3]	3
4		4
5		5
⋮	⋮	⋮
N-1		N-1
N		N

$1 + 2 + 3 + 4 + \cdots (N-1) + N$

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

Quiz- 9

```
for(i=1; i≤N; i++){
    for(j=1; j≤2^i; j++){
        ----------
    }
}
```

i	j	iterations for inner loop
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

total iteration = $2^1 + 2^2 + 2^3 + \dots + 2^N$

$= \frac{2\left[2^N - 1\right]}{(2-1)} = 2\left[2^N - 1\right]$

G.P

$a = 2$

$r = 2$

$N = N$.

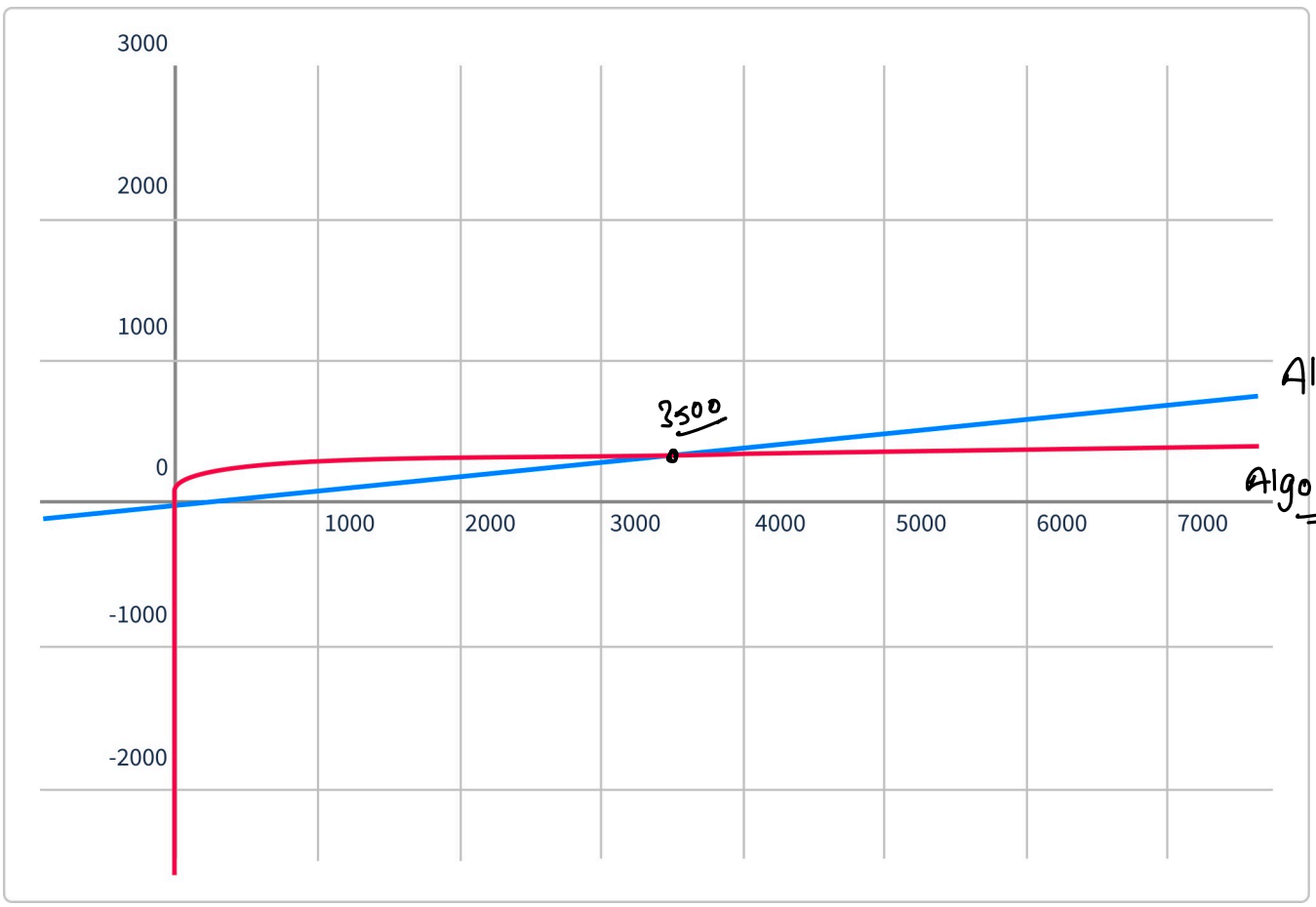
sum $\rightarrow \frac{a(r^N - 1)}{(r-1)}$

Algo.1

$y = 100*\log N$

Algo.2

$y = N/10$



$N < 3500$ ⇒ Algo-2 will perform better than Algo 1.

$N > 3500$ ⇒ Algo-1 will perform better than Algo-2.

India vs Pak ⇒ 30 m

Most no. of views on youtube video → 14 B

∴ In real world, data is really huge & keeps on increasing.

Asymptotic analysis of Algorithms

⇒ Analysis of algorithms for really large inputs.

Big-0 notation

1. Calculate Iterations based on Input Size
2. Ignore Lower Order Terms
3. Ignore Constant Coefficients

$100 \log N \rightarrow O(\log N)$

$N/10 \rightarrow O(N)$

---

iterations. → $4N^2 + 3N + 1$

$4N^2$

$O(N^2)$

Comparison order

$\log_2 N < \sqrt{N} < N < N\log_2 N < N\sqrt{N} < N^2 < N^3 < 2^N < N! < N^N$

$4N^2 + 3N + 6\sqrt{N} + 6\log_2 N + 50 \rightarrow O(N^2)$

$4N + 8N\log N + 1 \rightarrow O(N\log N)$

$4N\log N + 8N\sqrt{N} + 10^6 \rightarrow O(N\sqrt{N})$

Why do we ignore lower order terms?

Iterations → $N^2 + 10.N$

N	$N^2 + 10.N$ (Total iterations)	Percentage of 10.N in total iterations
10	$10^2 + 10 \times 10 = 200$	$\frac{100}{200} \times 100 = 50\%$
100	$100^2 + 10 \times 100 = 11000$	$\frac{1000}{11000} \times 100 \approx 9\%$
1000	$1000^2 + 10 \times 1000$	$\frac{10 \times 1000}{1000000} \times 100 \approx 1\%$

as input size ↑n, % of lower order terms in total iters ↓es.

Why to neglect co-efficient / constants?

Algo-1	Algo-2	
$10 \times \log_2 N$	N	→ Algo-1. is better
$100 \times \log_2 N$	N	→ " " "
$10^3 \times \log_2 N$	N	→ " " "
$9 \times N$	N^2	→ " " "
$90 \times N$	$\frac{N^2}{100}$	→ Algo-1 is better

rate of growth of N^2 >>> rate of growth of N.

Issues with Big-O

① $10^3 \cdot N$ — Algo-1 $\qquad$ N^2 — Algo-2

Algo-1 is always better than Algo-2 ? ✗

For larger inputs, Algo-1 is better than Algo-2 ? ✓

2. Can't compare when Big-O notation are same.

```
for(int i=1; i≤N; i++){
    if(i%2!=0){
        c=c+1;
    }
}
```

N iterations. → O(N)

```
for(int i=1; i≤N; i=i+2){
    c=c+1;
}
```

N/2 iterations → O(N)

Acc. to Big-O, both are same

but, actually second one will be better.

Online Editors and T.L.E

Amazon

1st Question → T.L.E. → think & implement → T.L.E → think & implement → T.L.E.

Online Servers → processing speed → 1 GHz

→ 1×10^9 instructions/sec.

↓

+, -, ×, /, =

==, <, >, function calling

variable declaration.

```
int countFactors(int N){
    count = 0;
    for(i = 1; i <= N; i++){
        if(N % i == 0){
            count++;
        }
    }
    return count;
}
```

// 1 iteration ≈ 6 instructions.

total Instructions = 6N + 3.

Approx-1

1 iteration → 10 instructions.

in 1 sec → 10^9 instructions are executed

in 1 sec → 10^8 × 10 instructions are executed.

in 1 sec → 10^8 iterations are executed

Approx-2.

1 iteration → 100 instructions

in 1 sec → 10^9 instructions are executed

in 1 sec → 10^7 × 100 instructions are executed.

in 1 sec → 10^7 iterations are executed

∴ **Conclusion**:

{ No. of iterations must be less than $10^7 - 10^8$, in order to submit the code. }

How should we approach a problem?

Read the problem statement

↓

Read the constraints carefully.

$1 \le N \le 10^5$

$\Rightarrow O(N^3)$ $(10^5)^3 \rightarrow 10^{15}$ iterations ✗

$\Rightarrow O(N^2)$ $(10^5)^2 \rightarrow 10^{10}$ iterations ✗

$\Rightarrow O(N)$ $10^5 \rightarrow 10^5$ iterations. ✓

$1 \le N \le 10^5$

↙ $N \log N$ ↘ N

$10^5 \log_2 10^5 \rightarrow 10^7$. ✓

N → perfect square

```
for(i = 1; i <= N; i++){
    if(i*i == N){
        return i;
    }
}
```

1
4
9
16
25
36

iteration → $\sqrt{N}$

250+ Problems.

↓

[for every problem discussed. T.C, S.C will be]