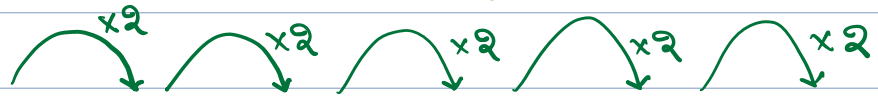
Agenda

1) Geometric Progression
2) Log Basics
3) Iteration Problems
4) Comparing Iterations using Graph
5) Time Complexity - Definition & Notations
6) Why do we get TLE ??

Geometric Progression (GP)



2, 6, 12, 24, 48, 96 .....

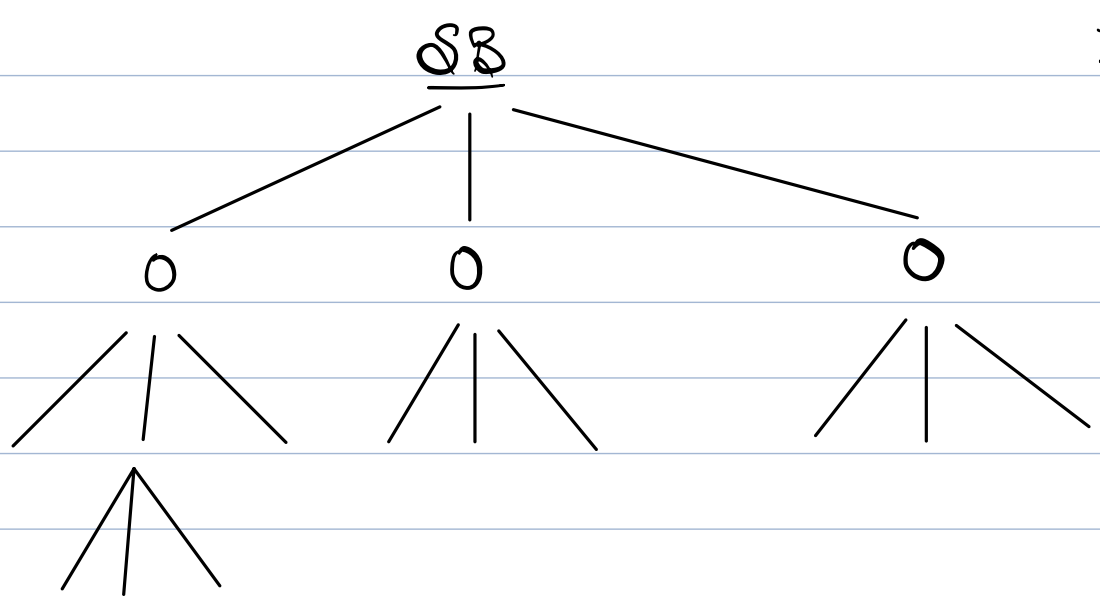
$(i-1)^{th}$ i

a (first term)

$\frac{48}{24} = 2 = r$ (common ratio)

Sum of first N terms of a GP

$$\frac{a \times (r^N - 1)}{(r - 1)} \quad (r \neq 1)$$



Logarithm Function

⇒ Inverse of exponential function

$\log_b a$ ⇒ log a to the base b

Eg $\log_b a = c$

$\Rightarrow b^c = a$

Eg $\log_2 8 = 3$

$$2^{3} = 8$$

Ex ⇒ 1) $\log_2 64 = 6$ $\left(\log_2 2^6\right)$

$$2^{6} = 64$$

2) $\log_5 25 = 2$

$$5^{2} = 25$$

3) $\log_2 10 = 3$

$2^3 = 8$
$2^{3.-} \quad 10$
$2^4 = 16$

4) $\log_2 40 = 5$

$2^5 = 32$
$2^{5.-} = 40$
$2^6 = 64$

NOTE:

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

$\Rightarrow K = \log_2 N$

★ $\log_a a^N = N$

Given a +ve no. N. How many times do we need to divide N by 2 (Consider only the integer part) to make it 1.

Eg N = 100 ⇒ 6

$100 \xrightarrow{1} 50 \xrightarrow{2} 25 \xrightarrow{3} 12 \xrightarrow{4} 6 \xrightarrow{5} 3 \xrightarrow{6} 1$

$9 \xrightarrow{1} 4 \xrightarrow{2} 2 \xrightarrow{3} 1$

Ans = 3

Sol^n $\log_2 N$

$$N \to \frac{N}{2^1} \to \frac{N}{2^2} \to \frac{N}{2^3} \ldots\ldots \text{(K times)} \to \frac{N}{2^K} = 1$$

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

$$N = 2^K$$

$$K = \log_2 N$$

I

$27 \Rightarrow \log_2 27 = 4$

$2^4 = 16$

27

$2^5 = 32$

$27 \to 13 \to 6 \to 3 \to 1$

Iterations

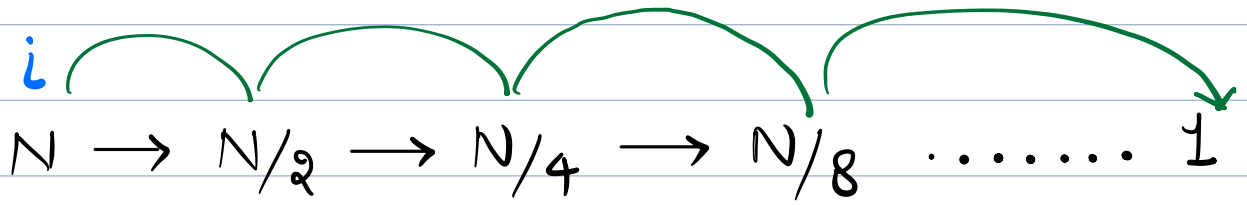
3

$N > 0$

$10 \rightarrow 5 \rightarrow 2 \rightarrow 1$

```
i = N
while ( i > 1) {

    i = i/2;
}
```



$\log_2 N$

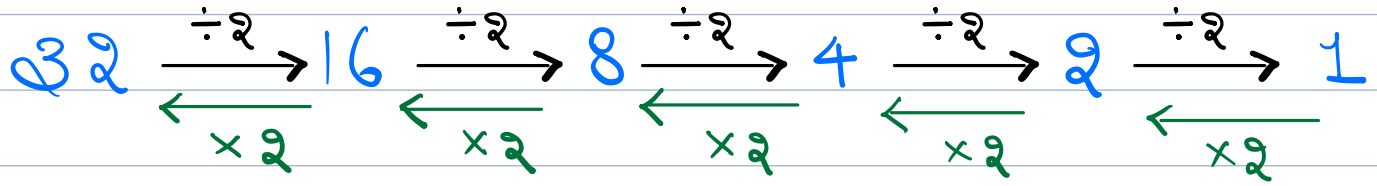
4

for (i goes from 1 to N-1 & gets multiplied by 2 in every iteration)

⇓

```
for ( i = 1 ; i < N ; i = i x 2) {
                ,,   ,,   ,,
}
```

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



$$Ans = \log_2 N$$

---

for (i goes from 0 to N-1 & gets multiplied by 2 in every iteration)

||

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

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

$N > 0$

$N = 15$

i

$0 \xrightarrow{\times 2} 0 \xrightarrow{\times 2} 0 \xrightarrow{\times 2} 0 \ldots\ldots \infty$

```
for (i = 1 to 10) {
    for (j = 1 to N) {
        mmmm
    }
}
```

i	j	# iterations
1	$[1, N]$	N
2	$[1, N]$	N
3	$[1, N]$	N
4	$[1, N]$	N
5		$\vdots$
$\vdots$		
9		
10	$[1, N]$	N

$10 \times N$

Q7

```
for (i = 1 to N) {          N
    for (j = 1 to N) { N
        //  //  //
    }
}
```

Ans = $N \times N$

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

$N \times N$

8

```
for (i = 1 to N) {
    for (j = 1 to N but multiplied by 2 every time) {
    }
}
```

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

i	j	# iterations
1	log N	log N
2	log N	log N
3	⋮	⋮
4	⋮	⋮
5	⋮	⋮
⋮	⋮	⋮

N^{N-1} $\log N$ $\frac{\log N}{N \times \log_2 N}$

Q9

```
for (i = 1 to 4) {
    for (j = 1 to i) {
        //
    }
}
```

i	j [1, i]	# iterations
1	[1, 1]	1
2	[1, 2]	2
3	[1, 3]	3
4	[1, 4]	4
		$\frac{4 \times 5}{2} = 10$

Q10

```
for (i = 1 to N) {
    for (j = 1 to i) {
        //
    }
}
```

i	j $[1, i]$	# iterations
1	$[1, 1]$	1
2	$[1, 2]$	2
3	$[1, 3]$	3
4	$[1, 4]$	4
⋮	⋮	
$N-1$	$[1, N-1]$	$N-1$
N	$[1, N]$	N
		$\frac{N \times (N+1)}{2}$

```
for (i = 1 to N) {
    for (j = 1 to 2^i) {
        // ... ...
    }
}
```

$2^i - 1 + 1 = 2^i$

i	j $[1, 2^i]$	# iterations
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	$[1, 2^{N-1}]$	2^{N-1}
N	$[1, 2^N]$	2^N

Add

$$2^1 + 2^2 + 2^3 + 2^4 \ldots\ldots 2^{N-1} + 2^N$$

$a = 2$

$r = 2$

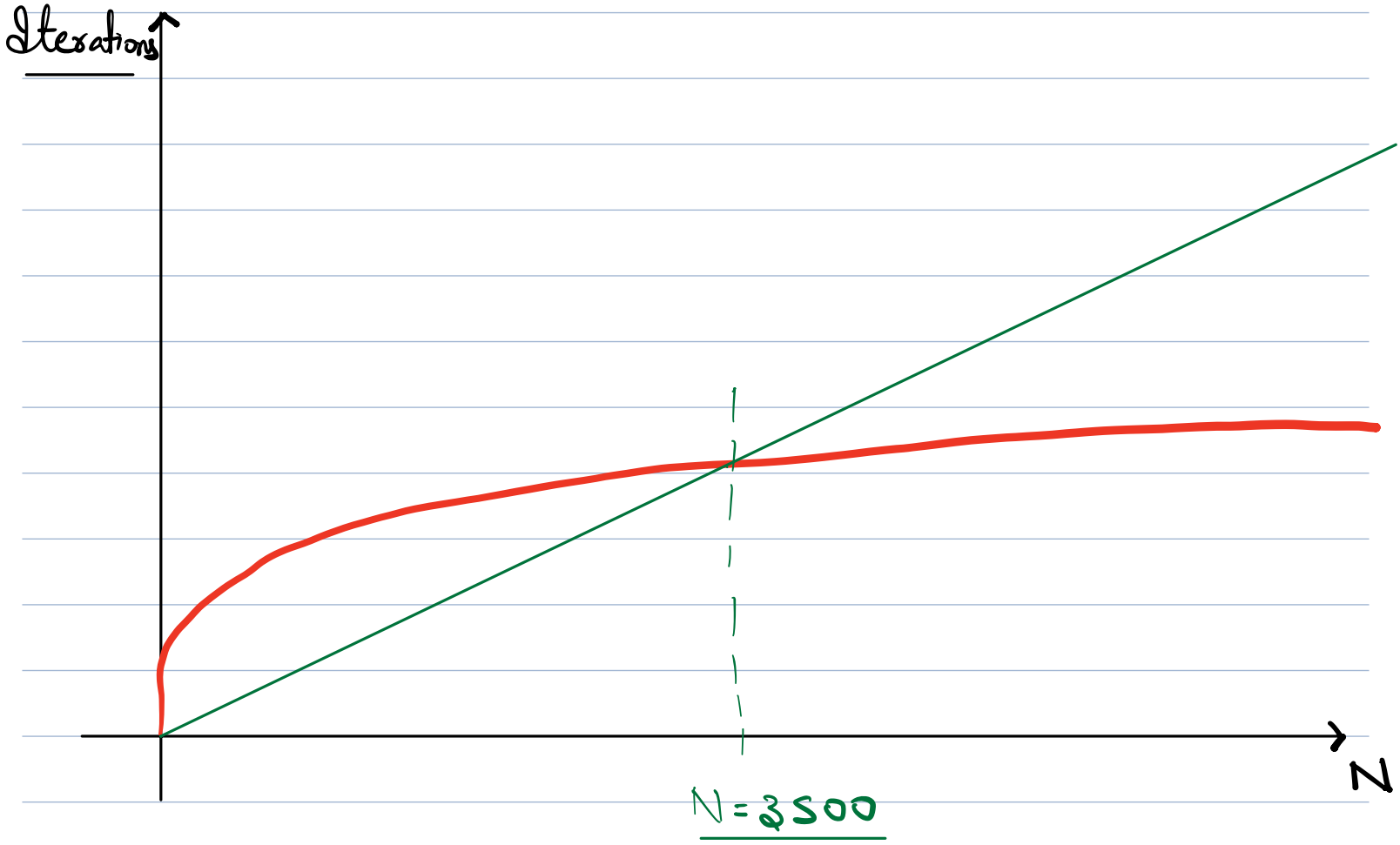
$$\frac{a(r^N - 1)}{(r-1)} = \frac{2(2^N - 1)}{(2-1)}$$

Ans = $2 \times (2^N - 1)$

Compare 2 Algo

Question → Sarika → $100 \times \log N$

→ Yash → $N/10$



for small I/P ($N < 3500$) ⇒ Yash

for large I/P ($N > 3500$) ⇒ Sarika

Ind V/s Pak Match ⇒ ≃ 30M (3 cr)

Youtube ⇒ Baby Shark ⇒ ≃ 15 Billion

⇒ The analysis of an algorithm when the I/P is huge is called as

Asymptotic Analysis

or

Big (O) notation } Time Complexity.

Time Complexity is the rate of growth of execution time w.r.t increase in input size.

Steps to calculate Big (O)

1) Calculate the no. of iterations based on I/P size
2) Ignore lower order terms
3) Ignore constant co-efficients of highest order term.

Eg 1 1) Iterations ⇒ $4N^2 + 3N + 1$

2) $4N^2 + \cancel{3N} + \cancel{1}$ (Ignore lower order)

3) $\cancel{4}N^2$ (Ignore constant co-efficients)

T.C. = $O(N^2)$

Comparison Order

$O(1) < \log(N) < \text{sqrt}(N) < N < N\log N < N \times \text{sqrt}(N) <$

Constant

$N^2 < N^3 < 2^N < N! < N^N$

Q 12

$\cancel{4N} + \cancel{3}\,N \times \log N + \cancel{1}$

T.C. = $O(N \times \log N)$

$$\cancel{4N\log N} + \cancel{3N\,\mathrm{sqrt}(N)} + \cancel{10^6}$$

$$= O(N\,\mathrm{sqrt}(N))$$

How can we Ignore Lower Order Terms ??

$$\text{Iterations} = \overset{H}{N^2} + \overset{L}{10N}$$

N	Total It.	Lower Order	% contribution
10	200 $(10^2 + 10\times10)$	100 (10×10)	50 %
100 (10^2)	$10^4 + 10^3$	10^3	$\frac{10^3}{10^4+10^3} \simeq 9\%$
10000 (10^4)	$10^8 + 10^5$	10^5	$\frac{10^5}{10^8+10^5} \simeq 0.1\%$

Issues with Big (O)

1) Does not compares for smaller I/P

2) For same higher order terms with different co-efficients, Big (O) considers them same

Algo 1

$4N^2 + 10N + 3$

~~$4N^2$~~ $+ 10N + 3$

$O(N^2)$

Algo 2

$100N^2 + 8N + 20$

~~$100N^2$~~ $+ 8N + 20$

$O(N^2)$

Doubt

```
for (i = 1 to N) {
    for (j = 1 to 3^i) {
    }
```

}

i	$j \in [1, 3^i]$	# iter
1		
2		
3		

$3 + 9 + 27 \quad \ldots \quad 3^{\wedge}$ times