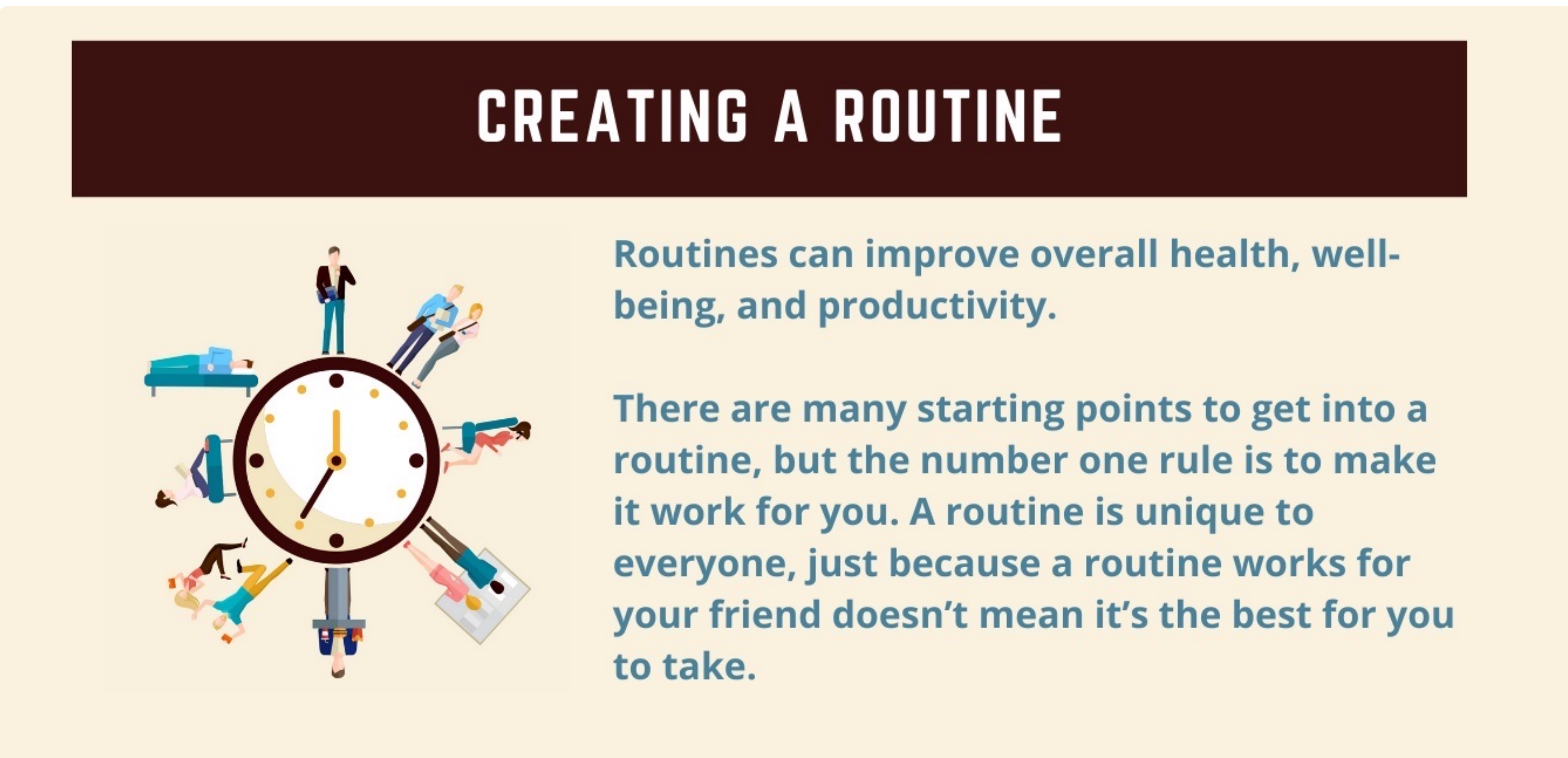
Agenda

1. Log Basics + Iteration Problems
2. Comparing Iterations using Graph
3. Time Complexity - Definition and Notations (Asymptotic Analysis - Big O)
4. Why do we get TLE ?



 SCALER

Log Basics - Logarithm is the Inverse of exponential Function.

$\log_b(a)$ - To what Value we need to raise b such that we get a.

$2^x = 64$

1. $\log_2 64 =$ $\log_2 2^6 = 6$

$3^x = 27$

2. $\log_3 27 = 3$

3. $\log_2 32 = 5$

4. $\log_2 10 = 3$

5. $\log_2 40 = 5$

$\boxed{\log_x x^y = y}$

6. $\log_2 2^6 = 6$

7. $\log_3 3^5 = 5$

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

N = 100	N = 324	N = 9	N = 27
$100 \to 50 \to 25 \to 12 \to 6 \to 3 \to 1$	$324 \to 162 \to 81 \to 40 \to 20 \to 10 \to 5 \to 2 \to 1$	$9 \to 4 \to 2 \to 1$	$\log_2 27$
ans :- 6	ans :- 8	ans :- 3	ans :- 4

No. of steps required to reduce N to 1 by repeatedly dividing with 2 = $\log_2 N$.

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

$$\frac{N}{2^0} \to \frac{N}{2^1} \to \frac{N}{2^2} \to \frac{N}{2^3} \to \cdots \to \frac{N}{2^K}$$

$2^0 = 1$

$3^0 = 1$

$\boxed{a^0 = 1}$

$4^0 = 1$

$5^0 = 1$

$$2^K = N$$

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

$$\boxed{K = \log_2 N}$$

$$\boxed{\log_x x^y = y}$$

Quiz- 1

$50 \rightarrow 25 \rightarrow 12 \rightarrow 6 \rightarrow 3 \rightarrow 1$

TC :- $O(\log N)$

$\log_2 N$

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

$N = 32$

$\log_2 N = \log_2 32$
$= 5$

Quiz- 2

$1 \rightarrow 2 \rightarrow 4 \rightarrow 8 \rightarrow 16 \rightarrow 32$

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

$N = 20$

$\log_2 16 = 4$

$1 \rightarrow 2 \rightarrow 4 \rightarrow 8 \rightarrow 16 \rightarrow 32$

Quiz- 3

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

	$i \le N$	$i = i*2$
$i=0$	$0 \le 5$	0
$i=0$	$0 \le 5$	0
$i=0$	$0 \le 5$	0
	⋮	

Infinite loop

SCALER

Quiz- 4

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

i	i ≤ 10	j
1	T	N
2	T	N
3	T	N
⋮	⋮	
10	T	N
11	F	

TC:- O(N)

Total No. of iterations $= N + N + N + \cdots + N$

$= 10N$

Quiz- 5

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

i	i ≤ N	j
1	T	N
2	T	N
⋮		
N	T	N
N+1	F	

Total No. of Iterations $= N + N + N + \cdots + N$

$= N * N$

$= N^2$

TC:- $O(N^2)$

SCALER

Quiz- 6

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

i	i ≤ N	j
1	T	log N
2	T	log N
⋮		
N	T	log N

No. of iterations = log N + log N + log N + . . . + log N
= N log N

TC:- O(N log N)

Quiz- 7

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

i	i ≤ 4	No. of j iterations
1	T	1
2	T	2
3	T	3
4	T	4
5	F	

No. of iterations = 1+2+3+4 = 10

TC:- O(1)

Quiz- 8

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

i	i ≤ N	No. of j iterations
1	T	1
2	T	2
3	T	3
4	T	4
⋮		
N	T	N

Total No. of iterations $= 1+2+3+4+\cdots N$

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

$$\frac{N^2+N}{2} = \frac{N^2}{2}$$

TC :- $O(N^2)$

Quiz- 9

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

i	i ≤ N	No. of j iterations
1	T	$[1, 2] - 2$
2	T	$[1, 2^2] - 4$
3	T	$[1, 2^3] - 8$
⋮		
N	T	$[1, 2^N] - 2^N$

No. of iterations $= 2+4+8+\cdots+2^N$

$a=2$

$r=2$

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

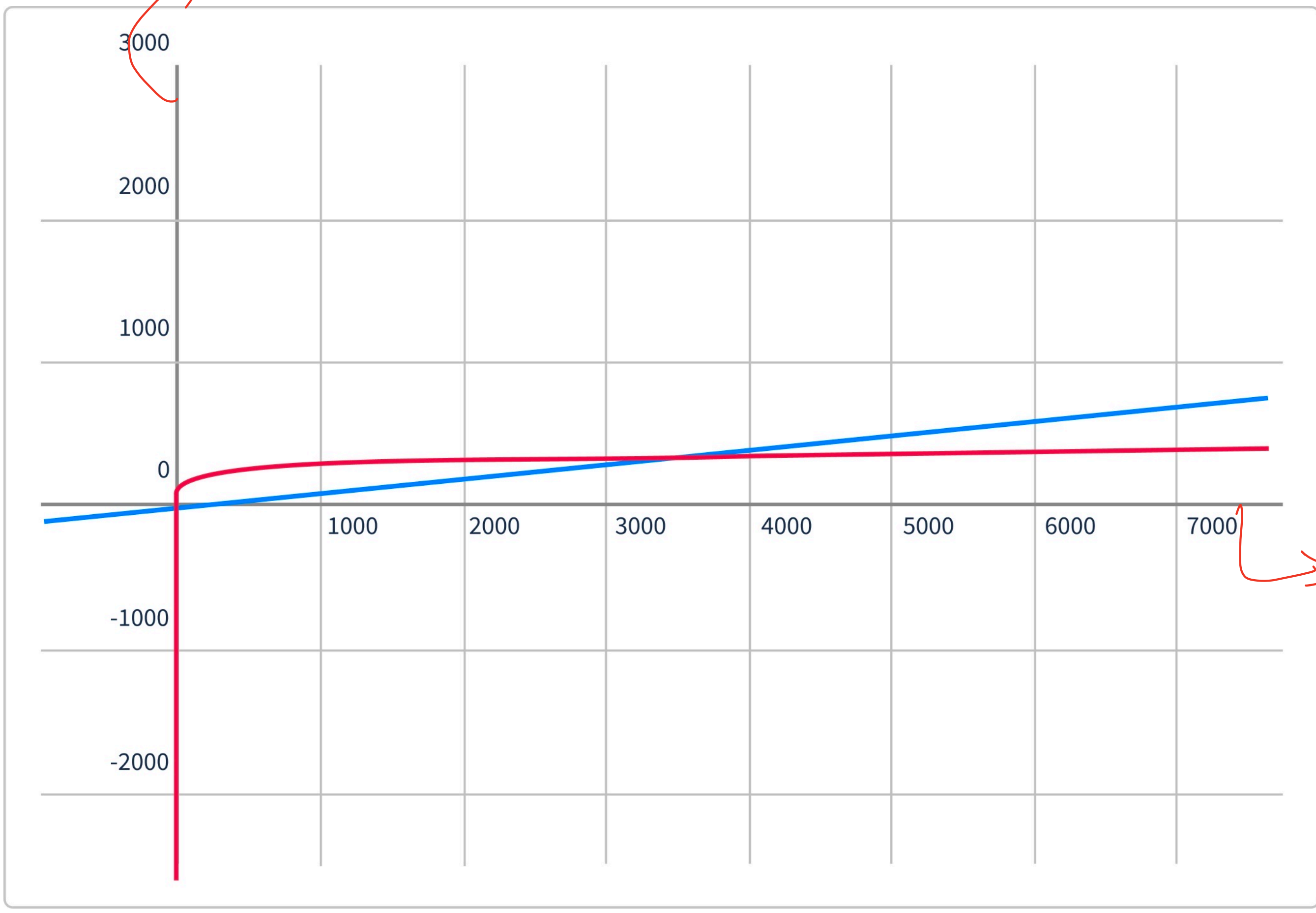
$$= \frac{2(2^n-1)}{2-1} = 2(2^n-1)$$

$TC = 2^n$

SCALER

Algo.1	**Algo.2**
100*logN	N/10

Execution time



N

N	Algo1	Algo2
1000	200	90
2000	210	150
3500	215	215
5000	230	300

N <= 3500, Algo 2 is Better

N > 3500, Algo 1 is Better

Asymptotic analysis of Algorithms

Asymptotic Analysis is used to estimate the performance of an Algorithm when the Input is large.

Big-0 notation

- Calculate the No. of iterations
- Ignore the lower order terms
- Ignore the constant coefficients.

Algo 1:- $100 \times \log N$

$O(\log N)$

Algo 2:- $\frac{N}{10}$

$O(N)$

Q. $4N^2 + 3N + 1$

$= 4N^2$

$= O(N^2)$

Comparision Order:-

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

$N = 36$

$5 < 6 < 36 < 180 < 216 < 36^2 < 36^3 < 2^{36} < 36! < 36^{36}$

Q. $4N^2 + 3N + 6\sqrt{N} + 9\log N + 10$

$= 4N^2$

$= O(N^2)$

Q. $4N + 3N\log N + 1 = 3N\log N$

TC:- $O(N\log N)$

Q. $4N\log N + 3N\sqrt{N} + 10^6$

TC :- $O(N\sqrt{N})$

Why do we ignore lower order terms?

10^3 of $10^4 + 10^3$

Iterations → $N^2 + 10.N$

N	$N^2 + 10.N$ (Total iterations)	Percentage of 10.N in total iterations
10	200	50%
100	$10^4 + 10^3$	≃ 9%
1000	$10^8 + 10^5$	0.1%

As the Input size Increases, the Contribution of lower order terms decreases

Why to neglect co-efficient / constants?

Algo 1	Algo 2	Winner for larger Input
$10\log N$	N	Algo 1
$100 \times \log N$	N	Algo 1
$9N$	N^2	Algo 1
$10N$	$N^2/10$	Algo 1

Issues with Big (O) :-

Issue 1 :- We Cannot always Say that one Algorithm will always be better than the other algorithms.

Algo 1 $\rightarrow N$

Algo 2 $\rightarrow N^2$

Input (N)	Algo1 ($10^3 * N$)	Algo 2 (N^2)	optimized
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	$10^3 \cdot (10^3+1)$	$(10^3+1)(10^3+1)$	Algo 1
10^4	10^7	10^8	Algo 1

2. If 2 Algorithms have Same higher order terms, then Big O is not Capable of Identifying the algorithm with higher/lower No. of Iterations

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

No. of Iterations = N

TC :- O(N)

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

No. of Iterations = N/2

TC :- O(N)

Online Editors and T.L.E

TLE → Time Limit Exceeded Error

1. processing speed of the server is 1 GHz i.e 10^9 Instructions per sec
2. Code should be executed in one second.

No. of Instructions $> 10^9$ in your code ⇒ TLE

Instruction → Mul, div, functi, if, declaring a variable

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

Total No. of Instructions = 2 +

No. of Instructions per iteration = 7

Total No. of Instructions = 2 + 7N

Instruction	Iteration
10	1
10^9	10^8

If for every iteration, we have 10 instructions, then we can have at max 10^8 iterations.

If for every iteration, we have 100 instructions, then we can have at max 10^7 iterations.

Conclusion :-

In our code, we can have 10^7 to 10^8 iterations only.

If we have more than that, we will get TLE.

Doubts :-

5 Instruction, 10^9 Iterations $\rightarrow 5 * 10^9$

10 Instruction, 10^8 Iteration $\rightarrow 10^9$

100 Instruction, 10^7 Iteration $\rightarrow 10^9$