SCALER

Time Complexity

TABLE OF CONTENTS

Log Basics

* What is meaning of log?

⇒ log is the inverse of exponential function.

Q. How do we read the statement "$(\log_b^a)$" (log of a base b.)

→ To what value we need to raise b such that we can get a.

$$b^x = a \Rightarrow x = \log_b^a$$

1. $\log_2 64 =$ $2^x = 64 \Rightarrow x = 6$ $(2^6 = 64)$

2. $\log_3 27 =$ $3^x = 27 \Rightarrow x = 3$

3. $\log_2 32 =$ $2^x = 32 \Rightarrow x = 5$

4. $\log_2 10 =$ $2^1 = 10$ $2^3 < 10 < 2^4 \Rightarrow x = 3.xy2$

$x = 3$ (floor value).

5. $\log_2 40 =$ $2^1 = 40$ $2^5 = 32 < 40 < 2^6 = 64$

$x = 5$ $2^{5.xy2} = 40$

6. $\log_2 2^6 =$ $x = 6$ $\log_a^{a^n} \Rightarrow n.$

7. $\log_3 3^5 =$ $x = 5$

$\log_3^{1024} \Rightarrow 3^x = 1024$

$x \approx 6$

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

N = 100

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

⇒ 6 times

N = 324

324 $\xrightarrow{①}$ 162 $\xrightarrow{②}$ 81 $\xrightarrow{③}$ 40 $\xrightarrow{④}$ 20 $\xrightarrow{⑤}$ 10 $\xrightarrow{⑥}$ 5 $\xrightarrow{⑦}$ 2 $\xrightarrow{⑧}$ 1

= 8 times

Quiz 1:

N = 9

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

N → divide N by 2 untill reach at 1

$$N \rightarrow \frac{N}{2} \rightarrow \frac{N}{2^2} \rightarrow \frac{N}{2^3} \rightarrow \frac{N}{2^4} \quad \ldots 1$$

$$60 \rightarrow \frac{60}{2^1} \rightarrow \frac{60}{2^2} \rightarrow \frac{60}{2^3} \rightarrow \frac{60}{2^4} \quad \ldots 1$$

$60 \rightarrow 30 \rightarrow 15 \rightarrow 7 \rightarrow \ldots 1$

let assume we are taking K steps to reach 1.

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

$$K = \log_2^N$$

$N = 27 \qquad \log_2^{27} \simeq 4$

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

Quiz- 1

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

$i = N \rightarrow \frac{N}{2^1} \rightarrow \frac{N}{2^2} \rightarrow \frac{N}{2^3} \dots \dots 1$

k is no. of iterations to reach

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

$2^k = N, \quad k = \log_2 N$

Quiz- 2

$1 \xrightarrow{*2} 2 \xrightarrow{*2} 4 \xrightarrow{*2} 8 \xrightarrow{*2} 16 \xrightarrow{*} 32 \dots N$

$1 \rightarrow 2^1 \rightarrow 2^2 \rightarrow 2^3 \rightarrow 2^4 \rightarrow 2^5 \dots N \ (2^k)$

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

$N \rightarrow \frac{N}{2} \rightarrow \frac{N}{2^2} \rightarrow \frac{N}{2^3} \dots 1$

$\log_2(N)$

$64 \xrightarrow{/2} 32 \xrightarrow{/2} 16 \xrightarrow{/2} 8 \xrightarrow{/2} 4 \xrightarrow{/2} 2 \xrightarrow{/2} 1$

$1 \xrightarrow{*2} 2 \xrightarrow{*2} 4 \xrightarrow{*2} 8 \rightarrow 16 \xrightarrow{*2} 32 \xrightarrow{*2} 64$

Quiz- 3

N>=0

~~N<=0~~

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

i	i<=N	loop work	update
0	0<=N	✓	i=i*2=0*2=0
0	0<=N	✓	i=i*2=0*2=0
0	0<=N	✓	i=i*2=0*2=0
⋮	⋮	⋮	⋮

Infinite iteration.

Quiz- 4

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

i	flow of j	Iteration
i=1	j: [1,N]	→ N iteration
i=2	j: [1,N]	→ N iteration
i=3	j: [1,N]	→ N iteration
⋮		
i=10	j: [1,N]	→ N iteration
i=11	→ loop breaks.	

Total iteration = N+N+N+...+N (10 times)

= 10*N

Quiz- 5

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

i	flow of j	iteration
1	j: [1, N]	N iteration
2	j: [1, N]	N
3	"	N
⋮		
N	j: [1, N]	N

N+1 → loop breaks.

$= N + N + N + \underbrace{\dots + N}_{N \text{ times}}$

$= N * N$

$= N^2$

Quiz- 6

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

i	flow of j	Iteration
1	j=1 to N, j=j*2 →	log N
2	j=1 to N, j=j*2 →	log N
3	j=1 to N, j=j*2 →	log N
4	- - -	- -
⋮		
N	j=1 to N, j=j*2 →	log N

$= \log N + \log N + \log N \dots$ ($\log N$, N times)

$= N * \log N$

Quiz- 7

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

= 1+2+3+4

10

i	flow of j	Iteration
1	j: [1, 1]	→ 1
2	j: [1, 2]	→ 2
3	j: [1, 3]	→ 3
4	j: [1, 4]	→ 4

5 → loop breaks

Quiz- 8

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

= 1+2+3+...+N

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

i	flow of j	Iteration
1	j: [1, 1]	→ 1
2	j: [1, 2]	→ 2
3	j: [1, 3]	→ 3
⋮		
N	j: [1, N]	→ N

Quiz- 9

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

i	flow of j	Iteration
i = 1	[1, 2^1]	→ 2^1 Iteration
i = 2	[1, 2^2]	→ 2^2 Iteration
i = 3	[1, 2^3]	→ 2^3 Iteration
⋮		
i = N	[1, 2^N]	→ 2^N iteration
i = N+1	→ stop	

$$\text{Total Iter} = 2^1 + 2^2 + 2^3 + 2^4 + 2^5 + \ldots + 2^n$$

$$= 2 + 4 + 8 + 16 + 32 + 64 + \ldots + 2^n$$

$$\text{sum of GP} = \frac{a*(r^n-1)}{(r-1)}$$

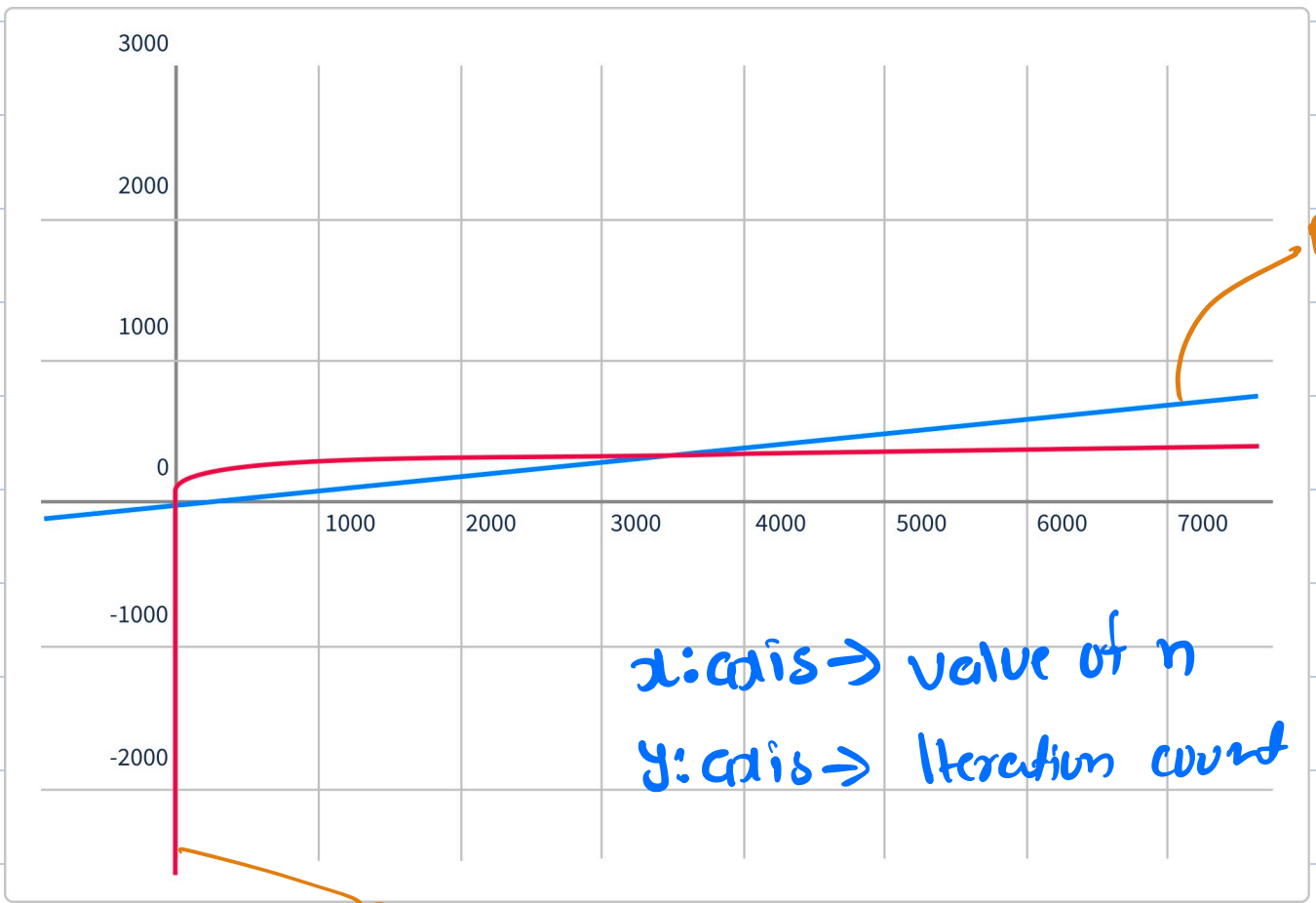
$a = 2$, Common ratio $(r) = 2$

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

$$= 2*(2^n-1)$$

Break: 10-32 pm

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



Algo2 is better than algo1 till 3500

But if n > 3500
Algo1 is better !

Whatsapk - 5 cr viewers

→ Baby shark video → 14 B views

Asymptotic analysis of Algorithms

→ Analysis of algorithm for larger input.

Calculation of Big-O notation:

Steps: 1) Calculate iteration based on input size.
2) Ignore lower order terms.
3) Ignore constant coefficient.

For eg.

$= 100 * \log_2^N$

100 → Consted Coefficient.

TC: $O(\log_2^N)$

$= \frac{N}{10}$

$= \frac{1}{10} * N$

$\frac{1}{10}$ → Constant

TC: $O(N)$.

$\Rightarrow 4n^2 + 3n + 1$

discard

$\Rightarrow 4n^2$

↳ constant

TC: $O(n^2)$

Comparision of order:

$O(1) < \log N < \sqrt{N} < N < N * \log N < N\sqrt{N} < N^2 < N^3 < N^4 < 2^N < N! < N^N$

from low to high

$\Rightarrow 4N^2 + 3N + 6\sqrt{N} + 9\log_2 N + 10$

lower order term

Steps:

1) Calculate iteration based on input size.
2) Ignore lower order terms.
3) Ignore constant coefficient

$\Rightarrow 4n^2$

$\Rightarrow O(n^2)$

Quiz: $F(n) = 4n + 3n\log(n) + 1$

$= 3n\log(n) + 4n + 1$

lower order

$= 3n\log(n)$ $= O(n\log n)$

↳ constant

Quiz: $F(n) = 4n\log n + 3n\,sqrt(n) + 10^6$

$= 3n\,Sqrt(n) + 4n\log n + 10^6$

lower order term

$= 3n\,sqrt(n)$

↳ Const $= O(n\sqrt{n})$

Why do we ignore lower order terms?

Iterations → $N^2 + 10.N$

value of n	total iteration	Iteration in lower order term	% of lower order iteration in total iteration
10	$= 10^2 + 10*10$ $= 200$	$= 10*10$ $= 100$	$\frac{100}{200} * 100 = 50\%$
100	$= 100^2 + 10*100$ $= 10^4 + 10^3$	$= 10*100$ $= 10^3$	$\frac{10^3}{10^4 + 10^3} * 100$ $\frac{10^3}{10^3(10+1)} * 100$ $= \frac{100}{11} \simeq 9\%$
10^5	$(10^5)^2 + 10*10^5$ $= 10^{10} + 10^6$	$10*10^5$ $= 10^6$	$\frac{10^6}{10^{10} + 10^6} * 100$ $\frac{10^6}{10^6(10^4+1)} * 100$

$\frac{10^4(10^4+1)}{2} \simeq \frac{10^8}{10^4}$ $\simeq 0.01\%$

Why to neglect co-efficient / constants?

Algo1 (Ajinky)	Algo2 (Tausif)	Winner Algo
$10 * \log n$	n	→ Ajinkya
$100 * \log n$	n	→ Ajinkya
$90 * n$	n^2	→ Ajinky (1)
$10 * n$	$\frac{n^2}{10}$	→ Ajinky (1)
$n \log n$	$100 * n$	→ Tausif

Conclusion: Winner always depend on variable but what if constant coefficient is so large.

⇒ Representing issue with Big O notation.

Issue with Big-O

Input Size	Algo1 ($10^3 n$)	Algo2 (n^2)	answer
100	$10^3 * 10^2$	10^4	Algo 2
10^3	$10^3 * 10^3 = 10^6$	$(10^3)^2 = 10^6$	same
10^3+1	$10^3(10^3+1)$	$(10^3+1)(10^3+1)$	Algo 1
10^4	$10^3 * 10^4 = 10^7$	$(10^4)^2 = 10^8$	Algo 1

Time Complexity

SCALER

Problem statement: Count even no. from 1 to n

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

Itr = n

O(n)

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

Itr = $\frac{N}{2}$

TC: O(N)

If two algo have same big O notation we can't decide which one is better.

Online Editors and T.L.E

⇒ Codechef, Codeforces, leetcode, gfg, interviewbit.

↳ Processing speed → 1 GHz

↓

10^9 instructions

Instruction Means, any operation such as multiplication, addition, variable declaration etc.

```
bool countFactor(N)
{
    int c=0;                    (1)
    for(i=1; i<=N; i++)         (2) (3) (4)
    {
        if(N%i==0)              (5) (6) (7)
        {
            c=c+1               (8)
        }
    }
}
```

approx.
We are doing 5/6 instruction per iteration

Suppose the code has 10 instruction per 1 iteration.

Instruction	Iteration
10	1
$10^8 * 10 = 10^9$	10^8

In one sec we can have at max 10^9 instruction or 10^8 iteration.

Iteration	=	100 instruction.
10^7	=	10^9

In general, code can have 10^7 or 10^8 iteration.

How should we approach a problem?

- Read the **Question** and **Constraints** carefully.
- Formulate an **Idea** or **Logic**.
- Verify the **Correctness** of the Logic.
- Mentally develop a **Pseudocode** or rough **Idea of Loops**.
- Determine the **Time Complexity** based on the Pseudocode.
- Assess if the time complexity is feasible and won't result in **Time Limit Exceeded (TLE)** errors.
- **Re-evaluate** the **Idea/Logic** if the time constraints are not met; otherwise, proceed.
- **Code** the idea if it is deemed feasible.

Q.

$1 <= N <= 10^5$

Big O will work or not ?

Complexity	Iteration	Works ?
$O(N^3)$	$(10^5)^3$	NO
$O(N^2 \log N)$	$(10^{10} \log 10^5)$	NO
$O(N^2)$	$(10^5)^2 = 10^{10}$	NO
$O(N)$	10^5	Yes