Time Complexity

- Log Basics
- Comparing Iterations using Graph
- Time complexity { Definition & Notations }
- TLE
- Importance of constraints.

Log Basics

$\log_{b}(a) = c$ — To what power should I raise b so that it equals a

base

$\Longrightarrow \quad b^c = a$

Examples

$\log_2(64) = 6 \quad \because \quad 2^{6} = 64$

c

$\log_3(27) = 3$

$\log_5(25) = 2$

$\log_2(32) = 5$

$\log_b(b^c) = c$

$\log_2(2^6) = 6$

$\log_3(3^3) = 3$

$\log_5(5^2) = 2$

$\log_2(2^5) = 5$

Q> Given a positive integer N, how many times do we need to divide by 2, until it reaches 1

Integer division $\frac{5}{2} = 2$

N = 10 $\quad \frac{10}{2} \longrightarrow \frac{5}{2} \longrightarrow \frac{2}{2} \longrightarrow 1 \quad 3$

N = 9 $\quad \frac{9}{2} \longrightarrow \frac{4}{2} \longrightarrow \frac{2}{2} \longrightarrow 1 \quad 3$

N = 30 $\quad \frac{30}{2} \longrightarrow \frac{15}{2} \longrightarrow \frac{7}{2} \longrightarrow \frac{3}{2} \longrightarrow 1 \quad 4$

N = 27 $\quad \frac{27}{2} \longrightarrow \frac{13}{2} \longrightarrow \frac{6}{2} \longrightarrow \frac{3}{2} \longrightarrow 1 \quad 4$

Given N, keep dividing by until we reach 1

$$\frac{N}{2} \longrightarrow \frac{N}{4} \longrightarrow \frac{N}{8} \quad \cdots\cdots\cdots \quad 1$$

Assume it will take k steps to reach 1

$$\frac{N}{2} \longrightarrow \frac{N}{2^2} \longrightarrow \frac{N}{2^3} \quad \cdots\cdots\cdots \quad \frac{N}{2^k}$$

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

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

Only take int value or $\boxed{\text{floor}(\log_2 N)}$

Floor of a value $\longrightarrow$ floor(1.5) $\longrightarrow$ 1

floor(2.2) $\longrightarrow$ 2

floor(2.99) $\longrightarrow$ 2

Iterations

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

iterations $\longrightarrow$ $\log(N)$

$$\underbrace{N \longrightarrow \frac{N}{2} \longrightarrow \frac{N}{2^2} \longrightarrow \cdots\cdots 1}_{\log_2 N \text{ steps}}$$

N

100

$$\frac{100}{2} \longrightarrow \frac{50}{2} \longrightarrow \frac{25}{2} \longrightarrow \frac{12}{2} \longrightarrow \frac{6}{2} \longrightarrow \frac{3}{2} \longrightarrow 1$$

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

$$\underbrace{1 \longrightarrow 2 \longrightarrow 4 \longrightarrow 8 \cdots\cdots \longrightarrow 2^k}_{\text{after } k \text{ steps we become or exceed } N} = N$$

$2^k = N$

$\log_2 2^k = \log N$

$k = \log N$

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

	i
step 1	0
step 2	0
step 3	0
⋮	⋮

$0 \longrightarrow 0*2 \longrightarrow 0*2^2 \ldots\ldots$

iterations $\longrightarrow \infty$

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

i	j	no. of iterations
1	[1, N]	N
2	[1, N]	N
⋮		⋮
10	[1, N]	N
11		$\boxed{10*N}$

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

i	j	no. of iterations
1	[1, N]	N
2	[1, N]	N
⋮		⋮
N	[1, N]	N

N times

iterations = N* N

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

i	j	no. of iterations
1	$\log_2 N$	$\log_2 N$
2	$\log_2 N$	$\log_2 N$
⋮		
N	$\log_2 N$	$\log_2 N$

N times

iterations = N* log N

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

i	j	# iterations
1	1	1
2	[1, 2]	2
3	[1, 3]	3
4	[1, 4]	4
		10

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

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

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

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

i	j	# iterations
1	[1, 2]	2
2	[1, 4]	4
3	[1, 8]	8
4		⋮
⋮		⋮
N	$[1, 2^N]$	2^N

GP

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

Comparing Iterations

	Sumit	Jahnavi
	Algo 1	Algo 2
Iterations $\longrightarrow$	$100 * \log(N)$	$N/10$

$N <= 3500$ Jahnavi's algo 2 was faster

$N > 3500$ Sumit's algo 1 was faster

Real world

world cup $\longrightarrow$ 5.6 Cr

Youtube $\longrightarrow$ 10+ Billion views

In real world the value of N is huge

Break : 22:38

Asymptotic Analysis / Big O

Analysing algorithm for large values of N.

Steps to calculate Big O

1> Calculate # of iteration.
2> Ignore lower order terms.
3> Ignore the coefficient.

# iterations	Big O
$100 * \log(N)$	$O(\log(N))$
$N/10$	$O(N)$
$\cancel{4}n^2 + \underbrace{\cancel{100n} + \cancel{\frac{\log N}{10}}}_{\text{step 2: ignore lower order terms}}$	$O(N^2)$

$1 < \log(N) < \text{sqrt}(N) < N < N\log(N) < N\,\text{sqrt}(N) < N^2 < N^3 < 2^N < N! < N^N$

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

// Drop lower order terms

$4N^2$

// Drop coefficient $\longrightarrow$ $O(N^2)$

Q> $4N + 3N\log N + 1$

step 2 $3N\log N$

step 3 $O(N\log N)$

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

which is higher order

$N\log N$ vs $N\sqrt{N}$ 64

$64 * 6$ vs $\boxed{64 * 8}$

higher order

step 2 $\longrightarrow$ $3N\sqrt{N}$

step 3 $\longrightarrow$ $O(N\sqrt{N})$.

Why do we ignore lower ordered terms ?

N	$N^2 + 10N$	% contribution of $10N$
10	$100 + 100$	$\frac{100}{200} * 100 = 50\%$
100	$10^4 + 1000$	$\frac{1000}{11000} * 100 \simeq 9\%$
10000	$10^8 + 10^5$	$\frac{10^5}{10^8} * 100 \simeq 0.1\%$

As N increases contribution of lower order terms $\downarrow$

Why ignore constant coefficient ?

Alok		Sujoy		Faster for large N
$10 \log(N)$	$O(\log N)$	N	$O(N)$	Alok
N	$O(N)$	$100 * \log N$	$O(\log N)$	Sujoy.
$9 * N$	$O(N)$	N^2	$O(N^2)$	Alok
$\frac{N^2}{10}$	$O(N^2)$	$10 * N$	$O(N)$	Sujoy
$N \log N$	$O(N \log N)$	$100 * N$	$O(N)$	Sujoy.

$2^{101} \log 2^{101}$ $\qquad$ $100 * 2^{101}$

$2^{101} * 101$ $\qquad$ $100 * 2^{101}$

lesser iteration $\longrightarrow$ faster code

Coeff doesn't matter for Big O

Limitations of Big O Notation

1) Algo 1 — Algo 2

$n^2 + 10n$ — $\frac{n^2}{10} + 100n$

$\underbrace{O(n^2) \qquad\qquad O(n^2)}_{\text{Both algos are same}}$

If Big O itself is same, compare the no. of iterations

2) Big O only works for very large values of N

	Algo 1	Algo 2	
N	$1000 N^2$	N^3	
10	10^5	1000	Algo 2
100	$1000 * 10^4$	10^6	Algo 2
1000	$1000 * 10^6$	10^9	same
10000	$1000 * 10^8$	10^{12}	Algo 1.

TLE Time limit Exceeded

Submit

TLE

Read ⟶ Think ⟶ pseudocode ⟶ code

Online editor ⟶ The server runs on 1GHz ....

10^9 instructions / second.

No. of instructions / iteration ~ 10 to 100

```
for {
    :
}
```
} 10 instruction / iteration

```
for {
    :
    :
    :
}
```
} 100 instruction

Time limit to run your code 1 second.

1 ⟶ 10^9 instructions
1 ⟶ 10^8 iterations
} small

1 ⟶ 10^9 instructions
1 ⟶ 10^7 iterations
} large

$10^7 \sim 10^8$

if your iterations are more than $10^7 \sim 10^8$
you get TLE

How to approach a problem

→ Read { description + constraints }.

→ Think

→ Pseudocode → Big (O)

N	Big (O)	iterations	
10^6	N^2	10^{12}	TLE
10^3	N^2	10^6	✓
10^4	N^2	10^8 → you don't know.	
10^9	$\sqrt{N}$	$3*10^4$	✓

N	Big O	iterations
10^6	N	10^6
10^4	N ✓, $N\log N$ ✓, $N\sqrt{N}$ ✓	10^4 → $10^4 * 12$, 10^6
10^3	N^2	$10^3 * 10^3 = 10^6$
10^2	N^3	$100*100*100=10^6$
20	2^N	$2^{20} \approx 10^6$

HW —— calculate Big O for all quizzes that we did.

Doubt session

Instruction — smallest calculation or step a CPU will do.

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

1 iteration $\longrightarrow$ 10 instructions } small code

100 instruction } large code

$n + 10^6$ $\quad$ n^2

$\longrightarrow$ i 1 to N

if $i*i == N$ return

$i == \sqrt{N}$ return

$\sqrt{N}$ iteration