

Logarithms

$$\log_b(a) = c \Rightarrow b^c = a$$

$$\log_2(64) = 6$$

$$2^6 = 64$$

$$\log_5(25) = 2$$

$$5^2 = 25$$

$$\log_2(10) = 3. \dots$$

$$\underset{8}{2^3} < 10 < \underset{16}{2^4}$$

$$\text{int}(\log_2(40)) = \underline{5}$$

$$2^5 < 40 < 2^6$$

Q → Given a +ve integer N, how many times we need to divide it by 2 to reach 1.

(integer division) $\frac{5}{2} = 2$

eg → 50

$$50 \rightarrow 25 \rightarrow 12 \rightarrow 6 \rightarrow 3 \rightarrow 1 \quad \text{Ans} = \underline{5}$$

$$9 \rightarrow 4 \rightarrow 2 \rightarrow 1 \quad \text{Ans} = \underline{3}$$

$$27 \rightarrow 13 \rightarrow 6 \rightarrow 3 \rightarrow 1 \quad \text{Ans} = \underline{4}$$

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

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

→ After K steps

$$\Rightarrow N = 2^K$$

$$\Rightarrow \boxed{\log_2(N) = K}$$

No. of iterations

1) $// N > 0$

$i = N$

#iterations = $\log_2(N)$

while ($i > 1$) {

TC = $O(\log(N))$

$i = i/2$

}

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

2) for ($i = 1$; $i < N$; $i = i * 2$) {
:
}

$i = 1 \rightarrow 2 \rightarrow 2^2 \rightarrow 2^3 \dots N$

#iterations = $\log_2(N)$ TC = $O(\log(N))$

3) $// N \geq 0$

for ($i = 0$; $i \leq N$; $i = i * 2$) {

:

$i = 0 * 2 = 0$

}

#iterations = ∞

4) for ($i = 1$; $i \leq 10$; $i++$) {
for ($j = 1$; $j \leq N$; $j++$) {
:
}

}

}

i	j	#iterations
1	[1 N]	N
2	[1 N]	N
⋮		

10	[1 N]	<u>N</u>
----	-------	----------

$$\underline{10 * N} \checkmark$$

$$TC = \underline{O(N)}$$

5) for (i=1; i <= N; i++) {
 for (j=1; j <= N; j++) {
 :
 }
 }

i	j	#iterations
1	[1 N]	N
2	[1 N]	N
⋮		
N	[1 N]	<u>N</u>

$$\underline{N * N} \checkmark$$

$$TC = \underline{O(N^2)}$$

6) for (i=1; i <= N; i++) {
 for (j=1; j <= N; j=j*2) {
 :
 }
 }

i	j	#iterations
1	1 - N	$\log_2(N)$
2	1 - N	$\log_2(N)$
⋮		⋮
N	1 - N	<u>$\log_2(N)$</u>

$$\underline{N * \log_2(N)}$$

$$TC = \underline{O(N \log(N))}$$

7) for (i=1; i <= 4; i++) {
 for (j=1; j <= i; j++) {
 :
 }
 }

<u>i</u>	<u>j</u>	<u>#iterations</u>
1	[1 1]	1
2	[1 2]	2
3	[1 3]	3
4	[1 4]	<u>4</u>

10 ✓

TC = O(1)

8) for (i=1 ; i <= N ; i++) {
 for (j=1 ; j <= i ; j++) {
 :
 }
 }

<u>i</u>	<u>j</u>	<u>#iterations</u>
1	[1 1]	1
2	[1 2]	2
⋮	⋮	⋮
N	[1 N]	<u>N</u>

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

TC = O(N²)

9) for (i=1 ; i <= N ; i++) {
 for (j=1 ; j <= 2ⁱ ; j++) {
 :
 }
 }

<u>i</u>	<u>j</u>	<u>#iterations</u>
1	[1 2 ¹]	2 ¹
2	[1 2 ²]	2 ²
⋮	⋮	⋮
N	[1 2 ^N]	<u>2^N</u>

2¹ + 2² + 2³ + ... + 2^N

$$a = 2 \quad r = 2$$

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

$$TC = \underline{O(2^N)}$$

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

Compare # iterations

Faisal

Algo 1

Mourika

Algo 2

iterations \rightarrow $100 * \log(N)$

$N/10$

$N \leq 3500$



$N > 3500$



Real World \rightarrow 1) live match views \approx 10 M

2) Highest watched youtube videos

\Rightarrow "Baby Shark" \approx 13 B

very large data

Asymptotic Analysis / Big O

Analysing performance of algorithms for large inputs.

Calculation of Big O \rightarrow Rate of growth of function

✓ 1) calculate # iterations wrt input size

\rightarrow 2) Ignore lower order terms

\rightarrow 3) Ignore constant coefficient

$$\text{eg} \rightarrow \# \text{iterations} \rightarrow (4N^2 + \cancel{3N} - 1) \rightarrow \underline{O(N^2)}$$

~~$4N^2$~~
 N^2

$$\rightarrow \log(N) < \sqrt{N} < N < N * \log(N) < N\sqrt{N} < N^2 < N^3 < 2^N < N! < N^N$$

$$2N^3 - 4\log(N) + 1500 * N\sqrt{N} \rightarrow \underline{O(N^3)}$$

Time complexity

#iterations

Rate of growth of ~~time~~ wrt input.

$TC = O(\# \text{iterations})$

$$O(4N + 3N\log(N) + 1) \rightarrow \underline{O(N\log(N))}$$

$$O(4N\log(N) + 3N\sqrt{N} + 10^6) \rightarrow \underline{O(N\sqrt{N})}$$

Why to neglect lower order terms?

N	#iterations ($N^2 + 10 * N$)	% contribution of lower order term
10	$10^2 + 10 * 10$ $= 100 + 100 = \underline{200}$	$\frac{100}{200} * 100 = \underline{50\%}$
100	$100^2 + 10 * 100$ $= 10000 + 1000$ $= \underline{11000}$	$\frac{1000}{11000} * 100 \approx \underline{9\%}$

10000	$10000^2 + 10 * 10000$ $= 10^8 + 10^5$	$\frac{10^5}{(10^8 + 10^5)} * 100 \approx \frac{10^7}{10^8} = 0.1\%$
-------	-------------------------------------------	----------------------------------------------------------------------

% contribution of lower order term is significantly low for large inputs.

Issues with Big O

1)

	<u>Algo 1</u>	<u>Algo 2</u>
# iterations →	$5000 * N$	N^2
TC →	$O(N)$ ✓	$O(N^2)$
$N = 1000$ →	$5 * 10^6$	10^6 ✓

Big O is only useful to understand algo w.r.t very large inputs.

2)

	<u>Algo 1</u>	<u>Algo 2</u>
# iterations →	$5000 * N$	$2 * N$ ✓
TC →	<u>$O(N)$</u>	<u>$O(N)$</u>

For algorithms where $O(-)$ is same, we have to check #iterations to decide which algo. is better.

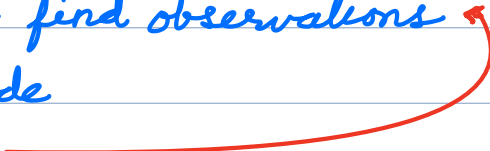
TLE (Time Limit Exceeded)

Amazon

Arur \rightarrow 2 questions in 1 hour

understand the question
find brute force
use examples to find observations
find optimised code
code it (TLE)

Expected TC \rightarrow Helps



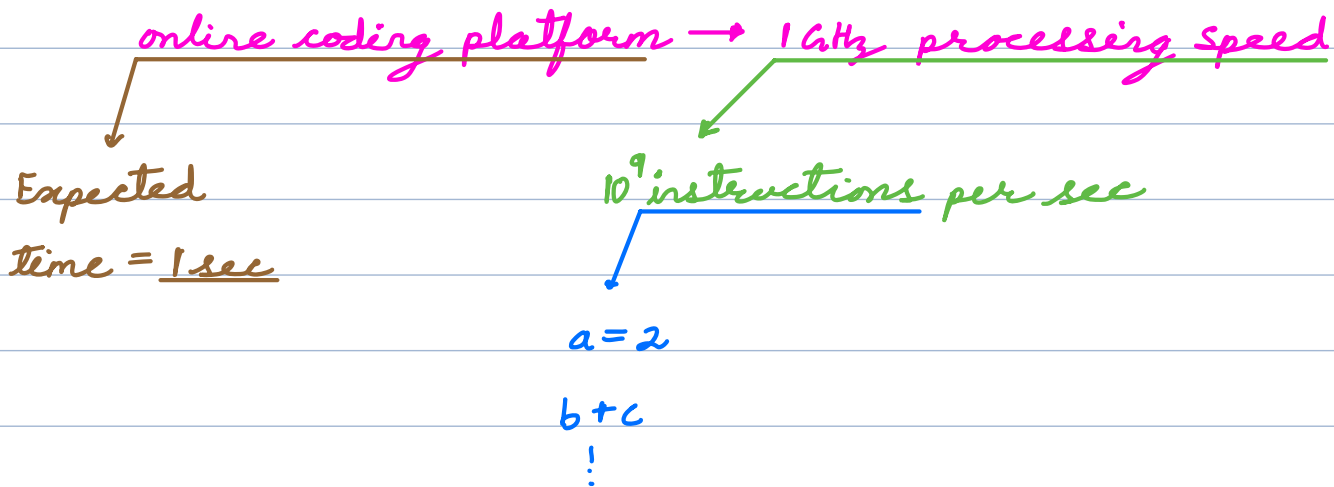
Why TLE occurs

online coding platform \rightarrow 1 GHz processing speed

Expected time = 1 sec

10^9 instructions per sec

$a = 2$
 $b + c$
!



let say 1 iteration takes 10 to 100 instructions.

10 instructions \rightarrow 1 iteration
 $\Rightarrow 10^9$ instructions $\rightarrow 10^8$ iterations

100 instructions \rightarrow 1 iterations

10^9 instructions $\rightarrow 10^7$ iterations

If time limit is 1 sec

then # iterations $\rightarrow 10^7$ to 10^8

$$TC = O(N^2)$$

$$N = 10^3$$
$$N = 10^5$$
$$N = 10^4$$

$$\rightarrow \underline{10^6} \quad \checkmark$$

$$\rightarrow \underline{10^{10}} \rightarrow TLE$$

$$\rightarrow \underline{10^8} \rightarrow \text{may or maynot give TLE.}$$

constraints

$$1 \leq N \leq 10^5$$

largest $N = \underline{10^5}$

$$\text{largest TC} = \underline{O(N \log N) / O(N) \dots}$$
