Log Basics

Iteration Problems

Comparing iterations using Graph

Time Complexity → Big O Notation

TLE

Importance of constraints

LOG

logarithm is inverse of power/exponents.

$\log_b a$ (log a to the base b)

↓

what should be power of b, so that we get a

$b^{?} = a$

$\log_2 64 = 6$ $\quad 2^6 = 64$

$\log_3 27 = 3$

$\log_5 25 = 2$

$\log_2 32 = 5$

$\log_2 10 = 3$

$2^{3.\_} = 10$

2^3 $\quad 2^4$

↓ $\quad$ ↓

8 — 10 — 16

$\log_2 8$ $\quad \log_2 16$

↓ $\quad$ ↓

3 $\quad$ 4

$\log_2 40 = 5$

$2^{5} \cdot - = 40$

2^5 ↓ 32 —— 40 —— 2^6 ↓ 64

① If $2^k = N \Rightarrow \log_2 N = k$

$\log_2 2^6 = 6$

$\log_3 3^5 = 5$

② $\log_a a^N = N$

N → how many times do we divide N by 2 till it reaches 1 ?

N = 100

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

ans = 6

N = 324

$324 \xrightarrow{/2} 162 \xrightarrow{/2} 81 \xrightarrow{/2} 40 \xrightarrow{/2} 20 \xrightarrow{/2} 10 \xrightarrow{/2} 5 \xrightarrow{/2} 2 \xrightarrow{/2} 1$

ans = 8

$N = 9$ $\qquad \log_2 9 = 3$

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

ans = 3

$N \xrightarrow{/2} N/2 \xrightarrow{/2} N/4 \xrightarrow{/2} N/8 \rightarrow \dots\dots 1$

$\frac{N}{2^0} \xrightarrow{/2} \frac{N}{2^1} \xrightarrow{/2} \frac{N}{2^2} \xrightarrow{/2} \frac{N}{2^3} \rightarrow \dots\dots$

(1) (2) (3)

After k steps $\frac{N}{2^k}$

$N \rightarrow 1$

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

$$\Rightarrow N = 2^k$$

Take $\log_2$ on both sides

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

$$\Rightarrow \log_2 N = k$$

$100 \xrightarrow{/2} \dots\dots 1$ $\qquad \log_2 100 = 6$

2^6 → 64 $\quad$ 2^7 → 128

27 $\quad$ $\log_2 27 = 4$

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

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

i_{bef}	Iteration	i_{aft}
N	1	N/2
N/2	2	N/4

$N \xrightarrow{1} N/2 \xrightarrow{2} N/4 \xrightarrow{3} N/8 \rightarrow \ldots\ldots \rightarrow 1$

After K steps, val of i = 1

$1 = N/2^K$

$N = 2^K$

$K = \log_2 N$ $\quad$ $O(\log_2 N)$

2. for (i=1 ; i<N ; i = i*2) {   (N = 32)

   ...

}

✓1 → ✓2 → ✓4 → ✓8 → ✓16 → ✗(32)

32 $\xrightarrow[①]{/2}$ 16 $\xrightarrow[②]{/2}$ 8 $\xrightarrow[③]{/2}$ 4 $\xrightarrow[④]{/2}$ 2 $\xrightarrow[⑤]{/2}$ ①

(×2 ×2 ×2 ×2 ×2)

$\log_2 N \rightarrow O(\log_2 N)$

$N \xrightarrow{/2} N/2 \xrightarrow{/2} N/4 \rightarrow \ldots\ldots 1$

3. $N >= 0$

for (i=0; i<=N ; i = i*2) {   (N = 32)

   ...

}

✓0 → ✓0 → ✓0 → ✓0 → 0 → ...

Infinite Iterations

4.

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

N = 5

[a b]

↓

b - a + 1

[1 N]

N - 1 + 1 = N

i	j	Iterations
1	[1 N]	N +
2	[1 N]	N +
3	[1 N]	N +
4		⋮
⋮		⋮
10	[1 N]	N +

$O(N)$ ← 10 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 +
3	[1 N]	N
⋮		⋮ +
N	[1 N]	N

N × N iterations

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

6.

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

i	j	Total Iterations
1	[1 N]	$\log_2 N$ +
2	[1 N]	$\log_2 N$ +
3	[1 N]	$\log_2 N$ +
⋮	⋮	⋮ +
N	[1 N]	$\log_2 N$
		$N \log_2 N$

$O(N \log_2 N)$

$j = 1$

① | ×2 ↓

2

② | ×2 ↓

④

③ | ×2 ↓

8

4 | ×2 ↓

16

⋮

N

$1 \xrightarrow{\times 2} 2 \xrightarrow{\times 2} 4 \rightarrow \cdots N$

$N \xrightarrow{/2} \cdots\cdots 1$

6.

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

$1 \xrightarrow{①} 0 \xrightarrow{②}$

i	j	Ite
1		

7. `for (i=1 ; i<=4 ; i++)` ← N
   `for (j=1 ; j<=i ; j++)`
   ...

i	j	Iterations
1	[1 1]	1 +
2	[1 2]	2 +
3	[1 3]	3 +
4	[1 4]	4
5	X	

10

i	j	Iterations
1	[1 1]	1 +
2	[1 2]	2 +
3	[1 3]	3 +
⋮	⋮	⋮ +
N	[1 N]	N

$N(N+1)/2 \rightarrow O(N^2)$

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

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

highest order term

$N^2/2$

$O(N^2)$

$10 N^0$

N^0

$O(1)$

8.

9.

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

i	j : [1 2^i]	Iterations
1	[1 2^1]	2^1 +
2	[1 2^2]	2^2 +
3	[1 2^3]	2^3 +
⋮		⋮
N	[1 2^N]	2^N

Total iterations =

$2^1 + 2^2 + 2^3 + \ldots + 2^N$

Sum of n terms in GP $= \frac{a(r^N - 1)}{r - 1}$

$a = 2 \quad r = 2 \quad n = N$

$$\text{Sum} = \frac{2(2^N - 1)}{2 - 1} = 2(2^N - 1) \text{ iterations}$$

$\downarrow$

$2.2^N - 2$

$\downarrow$ Highest order

2.2^N

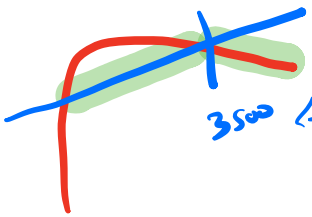
$\downarrow$ Ignore constant

$O(2^N)$

Compare two algorithms using iterations

↓

in terms of N



Red

3500 Algo1 (Kalu)

$100 \log_2 N$

Blue

Algo2 (Chitra)

$N/10$

Big(O)

$O(\log_2 N) < O(N)$

$O(\log N)$ is better than $O(N)$

	Iterations	Preferred
Till $N \leq 3500$	Chitra < Kalu	Chitra
$N > 3500$	Kalu < Chitra	Kalu

Lesser Iterations → Time

Data will increase over time

↓

Pick algo which works better for large input

Kalu's algo preferred for large input

tool → Desmos

Asymptotic analysis of algo

↓

analyse performance of algo on large input

↓

Big O Notation O( )

Steps for Big O

1. Calculate no. of iterations
2. Ignore lower order terms

| OR

Take highest order term → highest power of n

3. Ignore constant coefficient

$f(N) = 4N^2 + 3N + 1 \rightarrow 4N^2 + 3N^1 + 1N^0$

↓ Neglect lower order term

$4N^2$

↓ Remove constant coff

$O(N^2)$

Comparison order :

$\log_2(N) < \sqrt{N} < N < N\log N < N\sqrt{N} < N^2 < N^3 < 2^N$

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

$N = 36$

$5 < 6 < 36 < 36 \times 5$ (180)

$\log N < \sqrt{N}$

$N \log N < N\sqrt{N}$

$f(N) = 4N^2 + 3N + 6\sqrt{N} + 9\log_2 N + 10$

$4N^2 \rightarrow O(N^2)$

$f(N) = 4N^1 + 3N\log_2 N + 1\,N^0$

$3\,N\log_2 N$

$O(N\log_2 N)$

$f(N) = 4N\log N + 3N\sqrt{N} + 10^6 N^0$

$3N\sqrt{N}$

$O(N\sqrt{N})$

For constant iterations $\rightarrow O(1)$

① Neglect lower order terms

Algo (Tushar)

$F(N) = N^2 + 10N$

Input	Total iterations	% of lower order term in total iterations
10	200	$\frac{100}{200} \times 100\% = 50\%$
$100 = 10^2$	$10^4 + 10^3$	$\frac{10^3}{10^4 + 10^3} \times 100\% \simeq 9\%$
10^4	$10^8 + 10^5$	$\frac{10^5}{10^8 + 10^5} \times 100\% \simeq 0.1\%$

Obs: as input size ↑, contribution of lower order term in total iterations ↓

② Neglect constant coeff

Algo1 (Sandi)	Algo2 (Deepika)	For large inputs
$10 \log_2 N$ $10 \times \log_2 10^4$ 130	N 10^4 10^4	Sandi
$100 \log_2 N$	N	Sandi
$9N$	N^2	Sandi
$10N$	$N^2/10$	Sandi
$N \log_2 N$	$100N$ 10^9	Deepika
$10^9 \log 10^9$	10^{11}	

Issues with Big O notation

①
```
for (i=1 ; i<=N ; i++) {
    if (i%2 == 0) {
        c = c+1
    }
}
```
Iterations → N → O(N)

```
for (i=1 ; i<=N ; i=i+2) {
    c = c+1
}
```
↳ odd nos. from [1 N]

N = 10

1 → 3 → 5 → 7 → 9 → 11 (X)

Iterations → N/2 → O(N)

When 2 algo have same Big O notation, don't know which algo is better?

↓

Use no. of iterations

② Big(O)

$O(\log_2 N) < O(N)$

$O(\log N)$ is better than $O(N)$

	Iterations	Preferred
Till $N \leq 3500$	Chitra < Kalu	Chitra
$N > 3500$	Kalu < Chitra	Kalu

Kalu's algo is perfect for all input ? No

↓ (threshold point)
after a certain point

$N > 3500$ → kalu's

Doubts

① $10^9 \times \log_2 10^9$ $\quad 10^{11}$

$\log_2 10^9 = 9 \times \log_2 10$

$\log_2 n^n = n \log_2 n$ $\quad 9 \times 3.\_\_$

$= 27$

---

②

```
N
while (i > 5) {
    i = i/2
}
```

$N \xrightarrow{/2} N/2 \xrightarrow{/2} N/4 \cdots \cdots \rightarrow 5$

$-5 \qquad\qquad\qquad\qquad -5$

$N-5 \longrightarrow 0$

$\log_2 N-5$ iteration

int x = 8

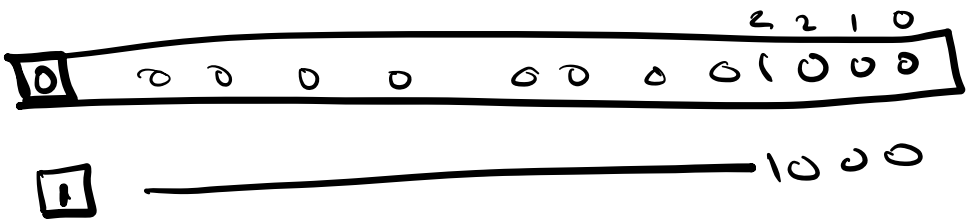
↓

4 B = 32 bits

double x = 8

↓

8 B = 64 bits



1

↓

20

$N \rightarrow \log_2 N + 1$

$8 \rightarrow \log_2 8 + 1$

$3 + 1$

Perfect No. → sum of smaller factors of A = A

```
sum = 0
for (i = 1 ; i < A ; i++) {
    if (A % i == 0)
        sum += i
}
if (sum == A)  perfect
```

6

↓ < 6

6 { 1, 2, 3

24

i

1 → 24

2 → 12

3 → 8

---

① $N^2 + 10N + 1 \rightarrow O(N^2)$

② $3N^2 + 100N \rightarrow O(N^2)$

$N^2 + 10N$? $3N^2 + 100N$

$N = 10^4$

$<$

$N = 10^6$

$<$

10^7

$<$

10^8