Agenda:

1. Log Basics
2. Iteration Problems
3. Comparing Iterations Using Graphs
4. Big O
5. TLE
6. Constraints

Logarithm:

$\log_b a = c \Rightarrow b^c = a \Rightarrow \underbrace{b \times b \times b \times \ldots \times b}_{c \text{ terms}} = a$

Eg: $\log_2 64 \rightarrow 6$

$\log_3 27 \rightarrow 3$

$\log_5 25 \rightarrow 2$

$\log_2 32 \rightarrow 5$

Calculate the floor value of the following

1. $\log_2 10 \rightarrow 3$

2. $\log_2 40 \rightarrow 5$

3. 3213

3.957 ⇒ 3

1. $\log_2(2^6) \rightarrow 6$

$a^x = a^N$

$x = N$

2. $\log_3(3^5) \rightarrow 5$

$\Rightarrow$ $\boxed{\log_a(a^N) = N}$

Question:

Given a positive integer N
how many times should we divide it
by 2 (consider only integer part)
till it becomes 1 ?

Eg: N: 100 $\xrightarrow{1}$ 50 $\xrightarrow{2}$ 25 $\xrightarrow{3}$ 12 $\xrightarrow{4}$ 6 $\xrightarrow{5}$ 3 $\xrightarrow{6}$ 1

$\log_2 100 = 6$

$\log_b a = c$

$\boxed{\log_2 9 = 3}$

N: 324

$\log_2 27 = 4$

$\log N$

N → N/2 → N/4 → . . 4 . 2 1

N ⟷ N/2 ← N/4 . . . . 4 2 1

Quiz 3:

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

$\Rightarrow \log_2 N$

$\Rightarrow \log N$

```
for (i=1 ; i<=N ; i=i*2)
{
    Print ("Something");
}
```

$\log N$

```
for (i=0 ; i<=N ; i=i*2)
{
    Print ("Something");
}
```

Infinite

```
for (i=1; i<=10; i++)
{
    for (j=1 ; j<=N; j++)
    {
        Print ("Something");
    }
}
```

$[0 \to N] \to N+1$

$10 * N$

```
for (i=0: i<=N; i++)
{   for (j=1; j<=N; j=j*2)
    {
        Print ("Something");
    }
}
```

$N+1$

$\log N$

$(N+1) \log N$

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

}
```

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

i	j
1	$[1,1] = 1$
2	$[1,2] = 2$
3	$[1,3] = 3$
4	$[1,4] = 4$
⋮	
N	$[1,N] = N$

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

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

# Iterations	Algo 1	Algo 2
	100 * log N (Purple)	N/10

$N <= 3500$

$N > 3500$

```
for(i → 1 to N)
{
    for(j → 1 to 2^i)
    {
        print;
    }
}
```

i
3

j
2

i	j
1	2
2	2
3	2

i	j
1	[1,2] → 2
2	[1,4] → 4
3	[1,8] → 8
⋮	
N	$[1, 2^N] \rightarrow 2^N$

$2 + 4 + 8 + \ldots + 2^N$ (N terms.)

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

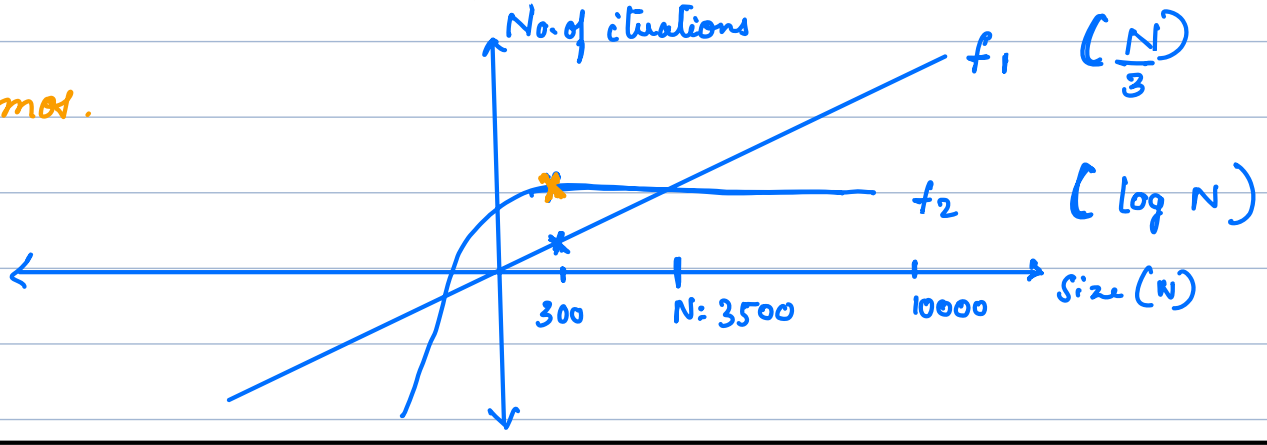
$= \frac{2(2^N - 1)}{1}$

$= 2(2^N - 1)$

Break till : 8:17 AM

Comparing iterations using graphs:

Desmos.



Asymptotic Analysis:

Analysing performance of algorithms for large inputs.

Calculation of Big O:

$2^N > N^2 > N\log N > N > \text{sqrt}(N) > \log N$

$A_1 \Rightarrow N^2$

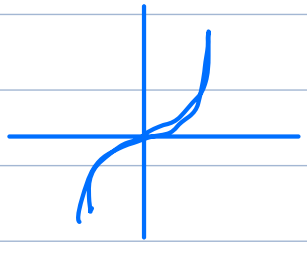
1 → Select highest ordered term (N)

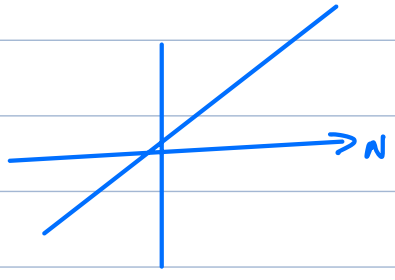
2 → Ignore constants.

$A_2 = 5\log N + 3N + 7$

$= O(N)$

$O(N^2)$ → Big O notation.





$F(N) = 4N + 3N\log N + 1$

$O(N\log N)$

$4N\log N + 3N\,\text{sqrt}(N) + 10^6$

$O(N\,\text{sqrt}(N))$

A_1	A_2
$N^2 + 3$	$4N^2$
$O(N^2)$	$O(N^2)$

Time Limit Exceeded:

TLE

1 GHz → 10^9 → instructions / sec.

10^8

```
x = x + 5 ;
```

```
for(i=0; i<10; i++)
{
    x = 5;    // 10
```

No. of iterations
* No. of instructions in 1 iterations.

10^8

$O(N)$

$1 \leq N \leq 10^{18}$

10^{18}

Understan
Brute Force
Observations → Optimise
Dry run
Code

Arrays.

Doubts:

outer : 3

inner : 2 → 2 →

inner : 3 → 3 →

$5 \times 3 = 15$

N

10^5 $O(N^2)$

$10^{10} \Rightarrow$ TLE

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

O(N)

```
fun ()
{
  fun2();
}
```

O(N²)

```
O(N²)
fun2()
for(  )
{
}
```

Constraint

$1 \leq N \leq 10^8$

Algo.

$O(N^2)$

$O(N)$

10^8

1 → 1 GHz → 10^9

10^8