Logarithms

$\log_b^a$ means to what power should we raise b to get a

$b^{?} \Rightarrow \log_b^a$

$b^{?} = a$

$\log_2^{64} = 6$ $\quad 2^{?} = 64$

$\log_3^{27} = 3$ $\quad 3^{?} = 27$

$\log_5^{25} = 2$ $\quad 5^{?} = 25$

$\log_2^{10} = 3$ $\quad 2^{?} = 10$ $\quad ? = 3.\text{rx}$

$2^3 = 8$

$2^4 = 16$

$\log_4^{27} = 2$ $\quad 4^{?} = 27$

$4^2 = 16$

$4^3 = 64$

$2^K = N \Rightarrow K = ?$

$b \overset{?}{=} a$ $\quad 2^K = N \Rightarrow K = \log_2^N$

$\Downarrow$

$2 \overset{?}{=} N$

$\log_2^N = K$

$7^K = N \Rightarrow K = \log_7^N$

$\log_2^{(2^6)} = 6$ $\quad 2 \overset{?}{=} 2^6 \Rightarrow$

$\log_3^{(3^5)} = 5$ $\quad 3 \overset{?}{=} 3^5$

$\log_a^{a^K} = K$ $\quad a \overset{?}{=} a^K$

Question: No. of times we have to divide N by 2 to make it 1

$N = 8 \xrightarrow[①]{/2} 4 \xrightarrow[②]{/2} 2 \xrightarrow[③]{/2} 1 \quad = 3$

$N = 16 \xrightarrow{/2} 8 \xrightarrow{/2} 4 \xrightarrow{/2} 2 \xrightarrow{/2} 1 \quad = 4$

$N = 100 \xrightarrow[1]{} 50 \xrightarrow[2]{} 25 \xrightarrow[3]{} 12 \xrightarrow[4]{} 6 \xrightarrow[5]{} 3 \xrightarrow[6]{} 1 \quad = 6$

$N = 9 \xrightarrow{/2} 4 \xrightarrow{/2} 2 \xrightarrow{/2} 1 \quad 3$

$N = 27 \xrightarrow{/2} 13 \xrightarrow{/2} 6 \xrightarrow{/2} 3 \xrightarrow{/2} 1 \quad 4$

$$N \xrightarrow{1} \frac{N}{2^1} \xrightarrow{2} \frac{N}{2^2} \xrightarrow{3} \frac{N}{2^3} \cdots\cdots\cdots \xrightarrow{k} \frac{N}{2^k}$$

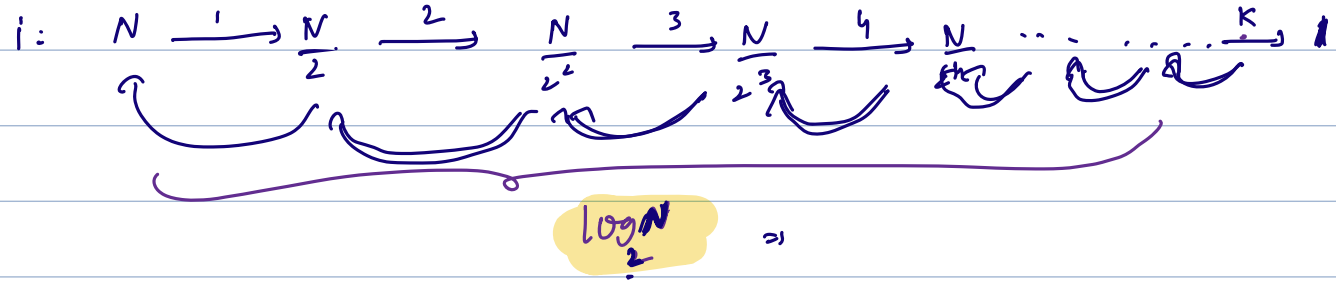
$$\frac{N}{2^k} = 1 \Rightarrow N = 2^k$$

$$\boxed{k = \log_2 N}$$

Quiz:

How many iterations will be there in this loop ?

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



Quiz:

How many iterations will be there in this loop

```
for(i goes from 1 to N-1 and gets multiplied by 2 in every iteration)
{
    ...
}
```

for (i = 1; i ≤ N-1; i = i * 2) {

}

i = 1 → 2 → 4 → 8 → ... → ≥ N

log N
2

$\log N \approx \log(N-1)$

for(i = 0; i ≤ N-1; i = i*2){

y

How many iterations will be there in this loop ?

```
N >= 0
for(i goes from 0 to N-1 and gets multiplied by 2 in every iteration)
{
    ...
}
```

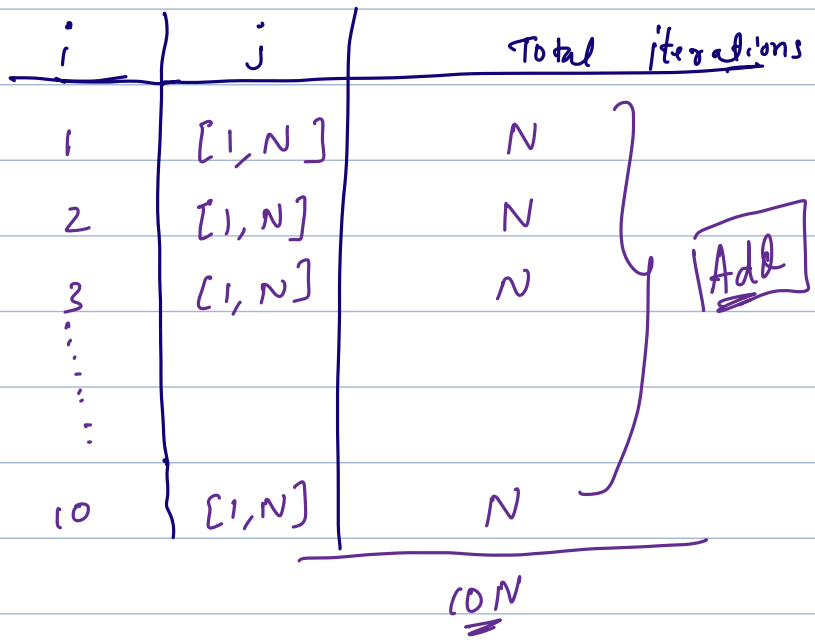
i before	Iteration #	i after
0	1	0
0	2	0
0	3	0
0	4	0
	!	
	;	
	'	
	'	
	'	
	'	

Infinite

Quiz:

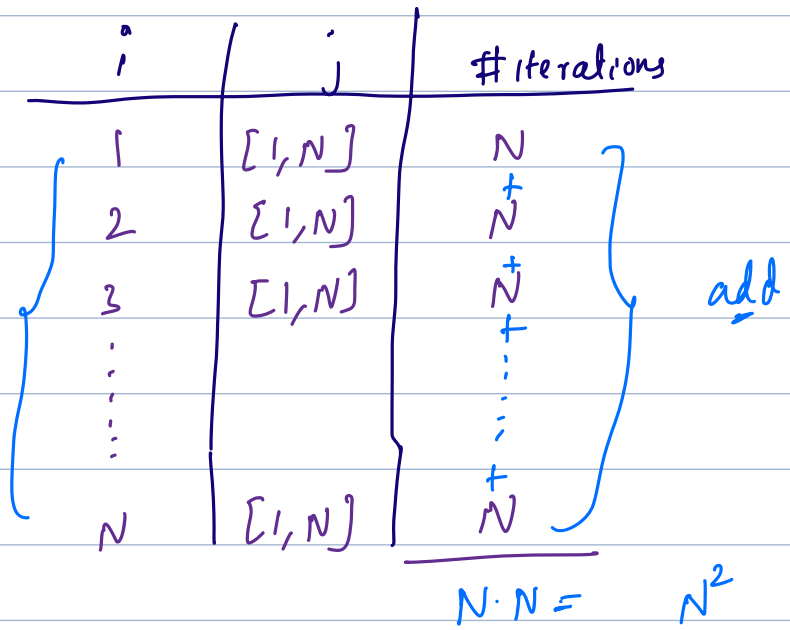
How many iterations will be there in this loop

```
for(i -> 1 to 10){
    for(j -> 1 to N){
    /       ..printf("Hey");
    }
}
```



How many iterations will be there in this loop

```
for(i -> 1 to N){
    for(j -> 1 to N){
        print("Hey");
    }
}
```



Quiz:

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

i	j	#iterations
1	$\log N_2$	$\log N_2$
2		$\log N_2$
3		$\log N_2$
⋮		
N		$\log N_2$

N times

$N \cdot \log N_2$

$[a, b] = b - a + 1$

Quiz:

```
for (i = 1; i <= N; i++) {
    for (j = 1; j <= i; j++) {
        print ("Hey");
    }
}
```

i	j	#iters
1	[1, 1]	1
2	[1, 2]	2 +
3	[1, 3]	3 +
4	[1, 4]	4 +
N-1	[1, N-1]	N-1 +
N	[1, N]	N

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

Quiz:

```
for (i = 1;  i ≤ N;  i++) {
    for (j = 1; j ≤ 2^i ; j++) {
        print ("Hey");

    }
}
```

i	j	#iters
1	$[1, 2^1]$	2
2	$[1, 2^2]$	2^2
3	$[1, 2^3]$	2^3
⋮		
N	$[1, 2^N]$	2^N

#iters = $2^1 + 2^2 + 2^3 + \cdots + 2^N$

$a = 2$

$r = 2$

$k = N$

$$Sum_k = \frac{a(r^k - 1)}{r - 1} = \frac{2(2^N - 1)}{2 - 1}$$

$$= 2(2^N - 1)$$

8:17am

Compare 2 Algorithms

	Divya (DivSort)	Teja (SortNumbers)
#Iterations	$100 \times \log N$	$\frac{N}{10}$

N: $[0, 3550] \Rightarrow$ Teja's Algo is better

$[3550, \infty] \Rightarrow$ Divya's Algo is better

We want to build Algos which work well for larger inputs

Asymptotic Analysis:
Analyzing the performance of algorithms for larger inputs.

Big-O

1) Find # of iterations (operations) [In terms of N]
2) Ignore lower order terms
3) Ignore any constant co-efficients

Ex1 : $11N^2 + 1000N + 10^6N^0 = O(N^2)$

Ex2 : $5N + 6N\log N + 1000N^0 = O(N\log N)$

$N \times \log N > N$

Ex3 : $4N + 3N\log N + 1 = O(N\log N)$

Ex4 : $4N\log N + 3N\sqrt{N} + 10^6N^0 = \boxed{N\sqrt{N}}$

~~$N \cdot \log N$~~ ~~$N\sqrt{N}$~~

$N = 2^{32}$

$= \log_2^{2^{32}}$ $\sqrt{2^{32}}$

$= 32 < 2^{16}$

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

Why to neglect lower order terms

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

N = 10

$(10)^2 + 10 \cdot 10 = 200$

N^2	$10N$	Contribution of lower order term
100	100	$\frac{100}{200} \times 100 = 50\%$

N = 100

N^2	$10N$	Contribution of lower order term
10^4	10^3	$\frac{10^3}{10^4 + 10^3} \times 100 = \frac{10^3 \times 100}{10^3(10+1)} \approx 10\%$

N = 10^4

N^2	$10 \cdot N$	%
10^8	10^5	$\frac{10^5}{10^8 + 10^5} \times 100 = \frac{10^5}{10^5(1000+1)} \times 100 = \frac{100\%}{1000} = 0.1\%$

As N ↑, contribution of lower order term almost becomes negligible

Why to ignore constant co-efficient?

Divya	Teja	For larger input
N	$10 \log N$	Teja
N	$1000 \log N$	Teja
$\frac{N}{100}$	$10000 \log N$	Teja

$N = 2^{32} \simeq 10^9$

$\frac{N}{100} = 10^7$

$10^4 \times \log_2 2^{32} = 32 \times 10^4$

Issues in Big - O

Issue 1:

Can we say Algo 1 is always better than Algo 2.

	Algo 1	Algo 2	
	$100N$	N^2	
	$O(N)$	$O(N^2)$	
$N = 10$	1000	100	Algo 2 is bettin
$N = 50$	5000	2500	Algo 2 is bett
$N = 100$	10^4	10^4	Both same
$N = 101$	100×101	101×101	Algo 1 is better

Issue 2:

Algo 1	Algo 2	
$10N^2 + 5N$	$5N^2 + 10N$	$\Rightarrow$ Algo 2 is better
$O(N^2)$	$O(N^2)$	

We are not able to Judge which Algo is better based on Big - O

$5N^2 + 10N + 3$ $5N^2 + 9N + 100$

Time Limit Exceeded Error (TLE)

How to avoid TLE? => Next class

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

$i \geq 2$

$N \to \frac{N}{2} \to \frac{N}{4} \to \cdots \to 4 \to 2 \to 1 \quad (\leq 1)$

$\log_2 N$

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

$y = 1 \to 2 \to 4 \to \cdots \to N$

$i \leq 5 \quad \Longleftrightarrow \quad i < 6$