

Log  $\Rightarrow$  inverse function to exponentiation.

$\log_b x \Rightarrow$  to what power should  
b be raised to reach x.

$$\log_b n = k$$

$$\boxed{b^k = n}$$

$$\# \log_2 4 = 2$$

$$2^2 = 4$$

$$\# \log_3 81 = 4$$

$$3^4 = 81$$

Ex1  $\log_2 8 \Rightarrow 3$

Ex2  $\log_2 2^{10} \Rightarrow 10$

$\log_2 n \Rightarrow k \Rightarrow$  no. of times n needs to  
be divided by 2 to reach  
1.

Ex1  $\log_2 16 \Rightarrow 4$

Ex2  $\log_2 32 \Rightarrow 5$

$\log_2 n \Rightarrow 2 \Rightarrow$  How many times should I multiply 1 by 2 to reach  $n$ .

# How many numbers in a range.

$$[3, 10] = 8$$

3, 4, 5, 6, 7, 8, 9, 10

$$[a, b] \Rightarrow b - a + 1$$

Both numbers  
are inclusive!

## Arithmetic Progression

→ Sequence of numbers where difference between consecutive terms is constant.

→ Constant difference is known as common difference ( $d$ )

Ex1 : 1, 4, 7, 10, 13, 16

Generalize :  $a, a+d, a+2d, a+3d, \dots, a+(n-1)d$

$n^{\text{th}}$  term  $\Rightarrow a + (n-1)d$

# Sum of 1<sup>st</sup>  $n$  terms =  $\frac{n}{2} [2a + (n-1)d]$

Ex1 : 1, 5, 9, 13

$a = 1, d = 4, n = 4$

$$\frac{4}{2} [2(1) + (4-1)4]$$

$$2 \times [14] = 28$$

## Geometric Progression

→ Sequence of numbers, where the next term can be found by multiplying the previous term by a fixed number.

$$\underline{\text{Ex1}}: 2, 4, 8, 16, 32$$

Common Ratio  $[r]$

$$\underline{\text{Ex2}}: 3, 12, 48, 192, 768 \dots$$

$$\underline{\text{Ex3}}: 1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16} \dots$$

$$\underline{\text{Generalize}}: a, ar, ar^2, ar^3$$

$$n^{\text{th}} \text{ term} = ar^{n-1}$$

$$\# \text{ Sum of } 1^{\text{st}} \text{ } n \text{ terms} = \frac{a(r^n - 1)}{r - 1}$$

Special Case:  $r < 1$  & series is infinite.

$$\text{then } r^n \Rightarrow 0.$$

$$\text{Sum} \Rightarrow \frac{a}{1-r}.$$

Q1 for (int i = 1 ; i ≤ n ; i++) L

$$S = S + i;$$

3

$$\Rightarrow i : [1, n]$$
$$n-1+1 = n \Rightarrow O(n)$$

Q2 for (int i=0 ; i <= 100 ; i++) L

\_\_\_\_\_

\_\_\_\_\_

3

$$i: [0, 100]$$
$$= 100 - 0 + 1$$
$$\Rightarrow 1 \circ 1 = 0(1)$$

time complexity  
is constant

Q3 for (int i=1; i ≤ n; i += 2) {

}

# n → 4 → 2 =  $\frac{n}{2}$

# n → 5 → 3 =  $\frac{n}{2}$

$$\left( \frac{n+1}{2} \right)$$

# n=4 →  $\frac{5}{2}$  → 2

# n=5 →  $\frac{6}{2}$  → 3

Integer Divisor.

$$\frac{n+1}{2} \Rightarrow \frac{n}{2} + \cancel{\frac{1}{2}}$$

$$= \underline{\underline{O(n)}}$$

Q4  $\text{for } (\text{int } i=1; i*i \leq n; i++)$   $\{$

$\{$

$$i^2 \leq n$$

$$i \leq \sqrt{n}$$

$$i = [1, \sqrt{n}]$$

$$\Rightarrow \sqrt{n} - 1 + 1$$

$$\Rightarrow \sqrt{n}$$

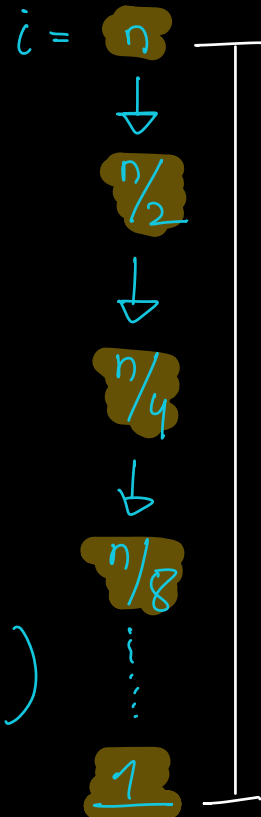
$$\Rightarrow O(\sqrt{n})$$

Q5

$i = n$   
while ( $i > 1$ ) {  
     $i = i/2$

}

no of iterations =  $\log_2 n$   
=  $O(\log n)$



Q6

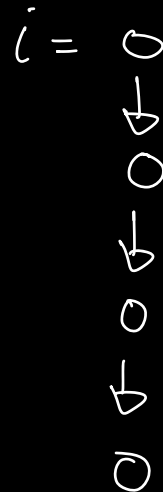
for (int i=0;  $i \leq n$ ;  $i = i * 2$ ) {

    //

}

in finite

TC  $\Rightarrow$   $O(\text{infinity})$





Q7 for (int i=1; i ≤ n; i=i\*2) {  
    }  
}

i = 1, 2, 4, 8, 16, 32, ..., n

|-----|

log N

→  $O(\log N)$

10:39

## NESTED LOOPS !

Q1 for (int i=1 ; i ≤ 10 ; i++) {

for (int j=1 ; j ≤ n ; j++) {

==

}

}

i	j	Total iteration
1	[1, n]	n
2	[1, n]	n
3	[1, n]	n
⋮		
10	[1, n]	n

10

10n

$O(n)$

⇒ 10 + 10n

int j = 0

for (int i = 0; i <  $10^5$ ; i++) {

while (j < 100) {

j++;

}

}

$O(1)$

i	j	Total
0	100	100
1	0	0
2	0	0
3	0	0
⋮		
$10^5 - 1$	0	0

$$\underline{10^5} + \underline{100}$$

Q2 for (int i=1 ; i ≤ n ; i++) {

for (int j=1 ; j ≤ n ; j++) {

→

}

}

i	j	
1	[1, n]	n
⋮		⋮
n		n

$O(n^2)$

$\underbrace{\quad}_n \rightarrow \underbrace{\quad}_{n^2}$

Q3 for (int i=1 ; i ≤ n ; i++) {

for (int j=1 ; j ≤ n ; j=j\*2) {

==

}

}

i	j	Total
1	log n	log n
2	log n	log n
⋮		
⋮		
n	log n	log n

$\overbrace{n} \quad + \quad \overbrace{n \log n}$

$O(n \log n)$

Qn

for (int i = 1 ; i ≤ n ; i++) L

for (int j = 1 ; j ≤ (2<sup>i</sup>) ; j++) L

⇒

3

3

H.W

Hint

G.P

How to calculate Big O from the number of iterations!

1) Ignore lower order terms

2) Ignore constants

Q1 no of iterations  $\Rightarrow 4n^2 + 3n + 2$

1<sup>st</sup> Step : Ignore lower order terms

$$4n^2 + \cancel{3n} + \cancel{2}$$

2<sup>nd</sup> Step : Ignore constants.

$$\cancel{4}n^2$$

$$\Rightarrow O(n^2)$$

Q<sub>2</sub> no of iterations  $\Rightarrow 3n\sqrt{n} + 4\log n + 3n\log n$

1<sup>st</sup> Step : Ignore lower order terms

$$3n\sqrt{n} + 4\cancel{\log n} + 3\cancel{n\log n}$$

n	$n\sqrt{n}$	$n\log n$
$2^{32}$	$2^{48}$	$2^{37}$
$2^{64}$	$2^{96}$	$2^{70}$

32

2<sup>nd</sup> Step : Ignore Constants.

$$\cancel{3n\sqrt{n}}$$

$$\underline{\underline{O(n\sqrt{n})}}$$





Double

for (int i=0 ; i < n ; i++) {

if (i == 2)

break;

}

for (int i=1 ; i ≤ n ; i+=2) {

}

i = 1, 3, 5, 7 ... n

$$a + (x-1)d \Rightarrow n$$

$$1 + (x-1)2 \Rightarrow n$$

$$(x-1)^2 \Rightarrow n-1$$

$$X-1 \Rightarrow \frac{n-1}{2}$$

$$X \Rightarrow \frac{n-1}{2} + 1 = \frac{n+1}{2}$$

# for (int i = 0 ; i < n ; i++)  $\mathcal{O}(n)$

$$\{ \text{for } (\text{int } i=0 ; i < m ; i++) \} \quad \text{I}_m$$

$$n + m$$

$$O(n+m)$$

$\{es \mid i=0; i \leq n; i++\}$  L

$\{es \mid i=0, j < i; j++\}$

}

i	j	count
0	0	0
1	1	1
2	2	2
3	3	3
n	n	n

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

$O(n^2)$

$$(2^0 + 2^1 + 2^2 \dots - 2^n)$$

$$\sim 2(2^n - \dots)$$

$$\approx \underline{\underline{O(2^n)}}$$