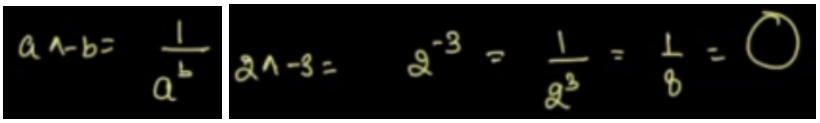
If you have any questions or suggestions regarding the notes, please feel free to reach out to me on WhatsApp: **Sanjay Yadav Phone: 8310206130**

Formulas ->

- Sum of natural numbers
  sn = N (N+1) / 2
- Inclusive bracket
  b - a + 1 = No. of iterations
- Exclusive bracket
  b - a = No. of iterations
- Common Ratio
  a * r / a = r (ratio) here **a** is first number and **r** is second number in list
  r / a = common ration **-** get common ratio
- Sum of N(natural numbers) term in the GP (Geometric progression)
  $S_n = a (r^n - 1) / r - 1$; a is first term (number) r common ration
- Get the 2 divide like N /2 / 2 - if value has to bata then we will do multiplication of below values **like**
  N / 2*2 = N/4;
- Prefix sum formula
  prefixSum[i] = prefixSum[i-1] + A[i]
- Keep in mind ceil, floor, pow will always return duble
- if power in minus then we will write like below.
  2 ^ -1 = ½^1 -> ½ = .5
  it means we will add batta of 1 and power in num below bata



Important Points:

- We always use a specific type of math called "logarithms" with a base of 2, instead of using different bases. This choice helps us keep things consistent.
- Sometimes, if we use a different base for logarithms, like $2e^2k$ and e, the results won't match. We want them to match, so we stick to using base 2 for our logarithms.
- When we add up numbers in a sequence, like 1 + 2 + 3 + ... + n, we have a special formula to find the answer: n(n+1)/2.
- Similarly, if we're adding numbers from 1 + 2 + 3 up to (n-1), we use a different formula: n(n-1)/2. We use these formulas to make our math easier and consistent.

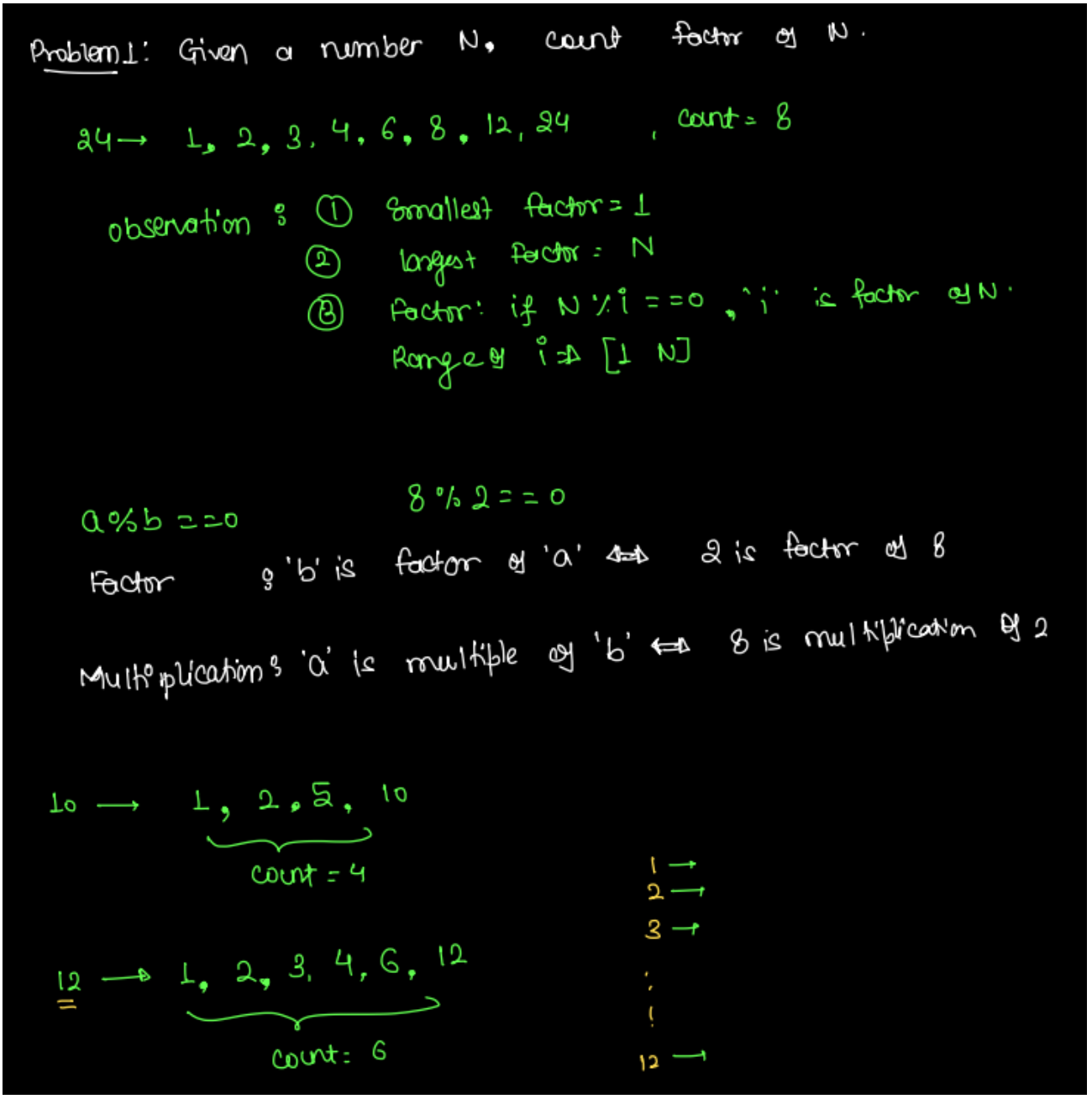
Comparison of Iterations

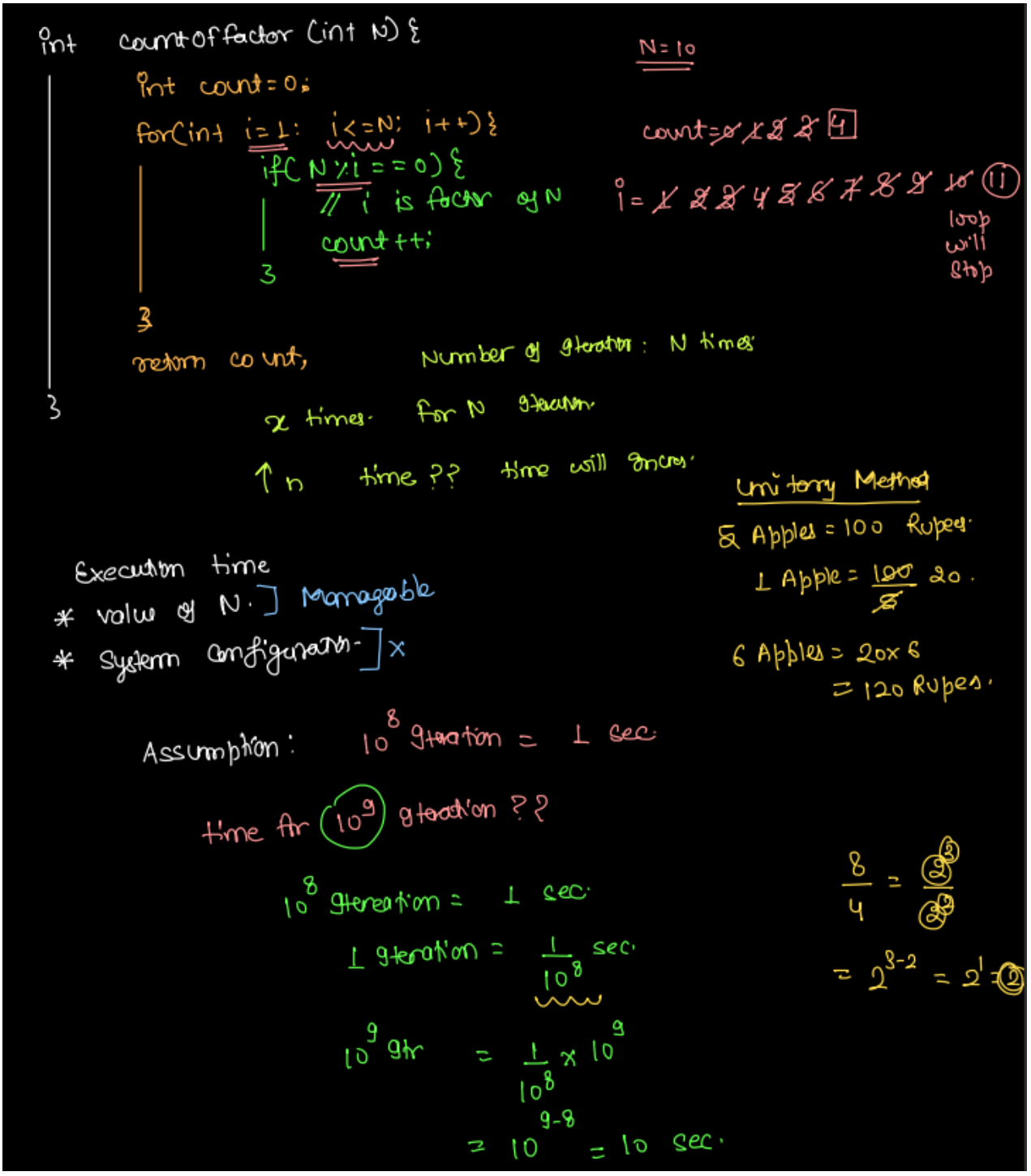
$$\log_2 N < \sqrt{N} < N < N\log_2 N < N\sqrt{N} < N^2 < 2^n$$

DSA: Introduction to Problem Solving - 18 - Apr 2023

- Count number of factor
- Improvisation in factor count
- Prime Number
- Reverse an array
- Reverse part of an array
- Rotate an array
- Log Basics

- **Count number of factor**





Improvisation in factor count

N	Iteration	time
10^8	10^8	1 sec.
10^9	10^9	10 sec
10^{18}	10^{18}	10^{10} sec.
		317 yrs

10^8 itr = 1 sec.

1 itr = $\frac{1}{10^8}$ sec

10^{18} itr = $\frac{1}{10^8} \times 10^{18}$ sec

$= 10^{18-8} = 10^{10}$ sec.

$\frac{N}{10^{18}}$

Iteration: $(\sqrt{N})$

$\sqrt{10^{18}} = (10^{18})^{1/2} = 10^{9 \times \frac{1}{2}} = (10^9) \equiv 10$ sec.

2-4 rebirth.

$\sqrt{a} = (a)^{1/2}$

$(a^b)^{1/2} = (a)^{b * \frac{1}{2}}$

Improvisation:

we have 2 number, i, j, N

$i * j = N$ { Both i and j are factor of N }

$3 * 4 = 12$ 3 & 4 both are factor of 12.

$\frac{a}{b} \times \frac{c}{d}$

$\left[\frac{3}{2} \times \frac{24}{6}\right] \Rightarrow a*d = b*c$

$6*8 = 24*2 \Rightarrow 48 = 48$

i → First number

$j = \frac{N}{i}$

First factor

N = 24

i	N/i
1	24
2	12
3	8
4	6
6	4
8	3
12	2
24	1

Part 1 | Part 2

Observation:

All factors are available in part 1.

Part 1:

$i <= \frac{N}{i}$

$\Rightarrow i * i <= N$

$\Rightarrow i^2 <= N$

taking Square Root both side

$\sqrt{i^2} <= \sqrt{N} \Rightarrow$

$i <= \sqrt{N}$

N = 100

i	N/i
1	100
2	50
4	25
5	20
10	10
20	5
25	4
50	2
100	1

Part 1 | Part 2

observations:

1. All the factors are available in first part
2. Range of first part i = 1 to $i <= \sqrt{N}$.

$\sqrt{4*4} = \sqrt{16} = \sqrt{4^2} = 4$

$\sqrt{15} \approx 3.*XYZ = 3$

N = 15

i	N/i	count	
1 ✓	15	2	{1, 15}
2 → X			
3 ✓	5	4	{3, 5}

(4) → loop stop

```
int countFactor(int N) {
    int count = 0;
    for(int i = 1; i <= sqrt(N); i++) {
        if(N % i == 0) {
            // i is factor, N/i is also a factor
            count += 2;
        }
    }
    return count;
}
```

count = (4)

$\begin{bmatrix} 1, 15 \\ 3, 5 \end{bmatrix}$

N = 16

$\sqrt{16} = 4$

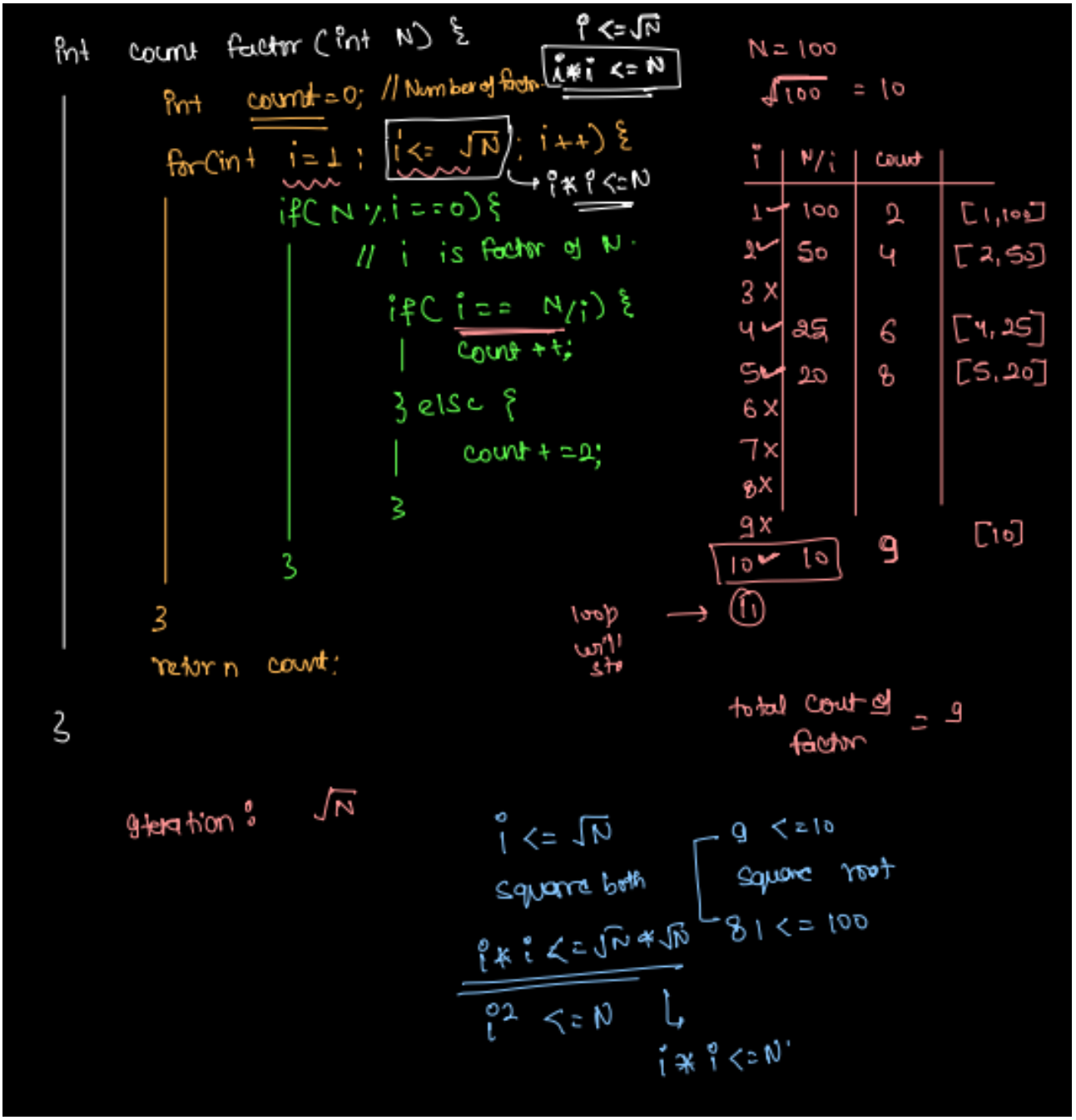
i	N/i	count	
1 ✓	16	2	[1, 16]
2 ✓	8	4	[2, 8]
3 X			
(4) ✓	(4)	6	

loop will stop → (5)

issue is: once i = 4, $\frac{N}{i} = 4$

both are same, that means we have single factor is there

observation: if $i == \frac{N}{i}$, increase count by 1 only



- **Prime Number**

Problem 2: Given a number N, check if number is prime or not?

Prime Number: If number have exactly two factor then number is prime.

Numbers:

15 = 1 3 5 15 → (4)

12 = 1 2 3 4 6 12 → (6)

9 = 1 3 9 → (3)

23 = 1 23 → (2) } ⇒ prime Number

count = (4)

Steps: (1) Calculate number of factor (optimised approach) for N.

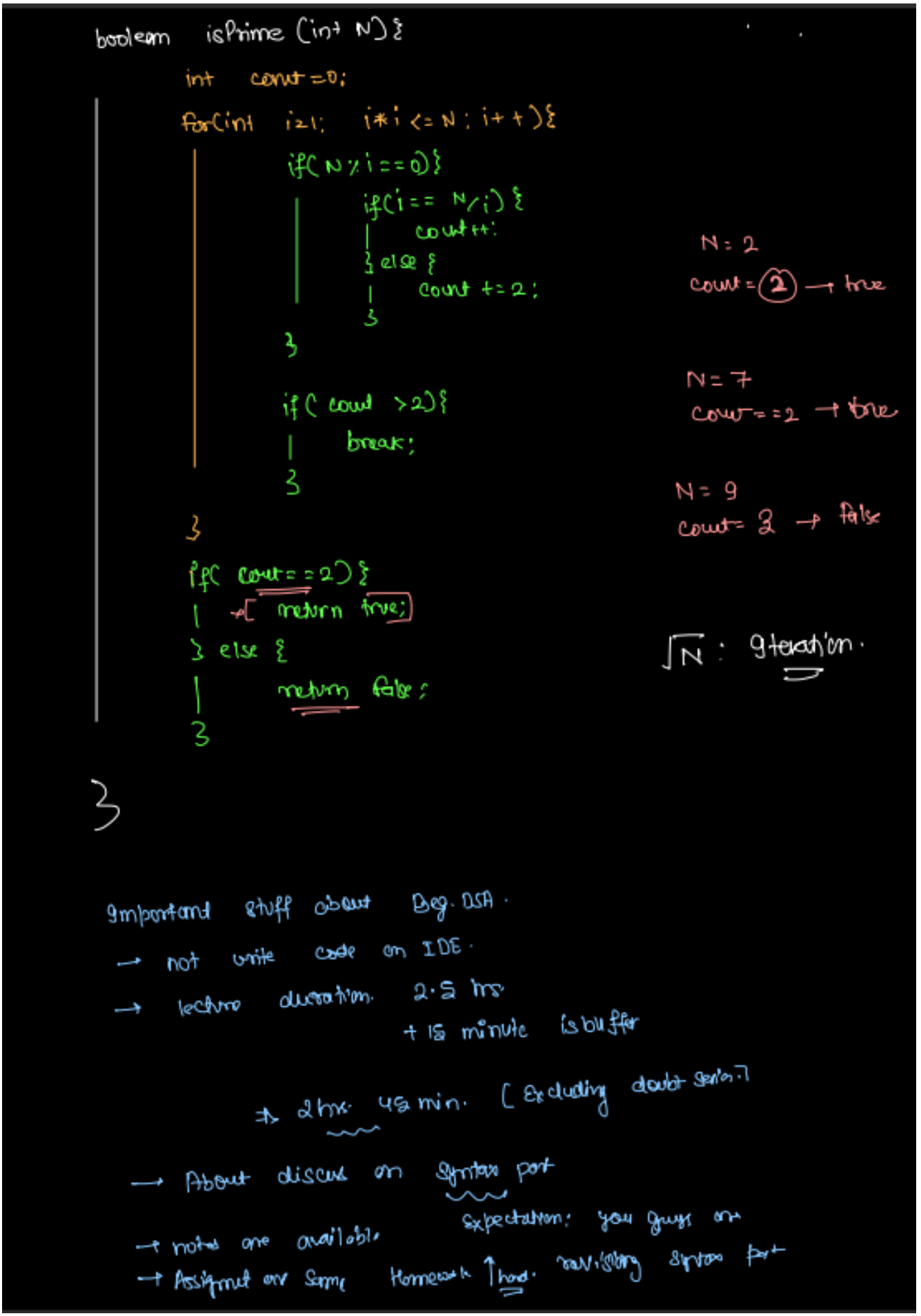
(2) If number of factor == 2, return true.

(3) otherwise return false.

factor count for N ⇒ $\sqrt{N}$ iteration.

1 to $\sqrt{N}$ → iteration.

Scope of improvement
→ if factor count > 2
break the loop.



- **Reverse an array**

Problem 8: Given an array A[]. reverse the array.

Constraint: ① Number of iteration allowed is N.

② Extra array is not allowed.

A: | 10 | 20 | 30 | 40 | 50 | 60 |
0 1 2 3 4 5

⇒ | 60 | 50 | 40 | 30 | 20 | 10 |
0 1 2 3 4 5

given an array and two index i1 and i2.
Swap the elements of i1 index and i2 index.

A: | 10 | ~~20~~ | 30 | 40 | ~~50~~ | 60 |
0 1 2 3 4 5

50 → 20

i1 = 1
i2 = 4

temp = 20

```
int temp = A[i1];
A[i1] = A[i2]
A[i2] = temp
```

i1 - index and i2 - index is swapped.

| ~~10~~ | ~~20~~ | 30 | ~~40~~ | ~~50~~ |
0 1 2 3 4

50 40 20 10

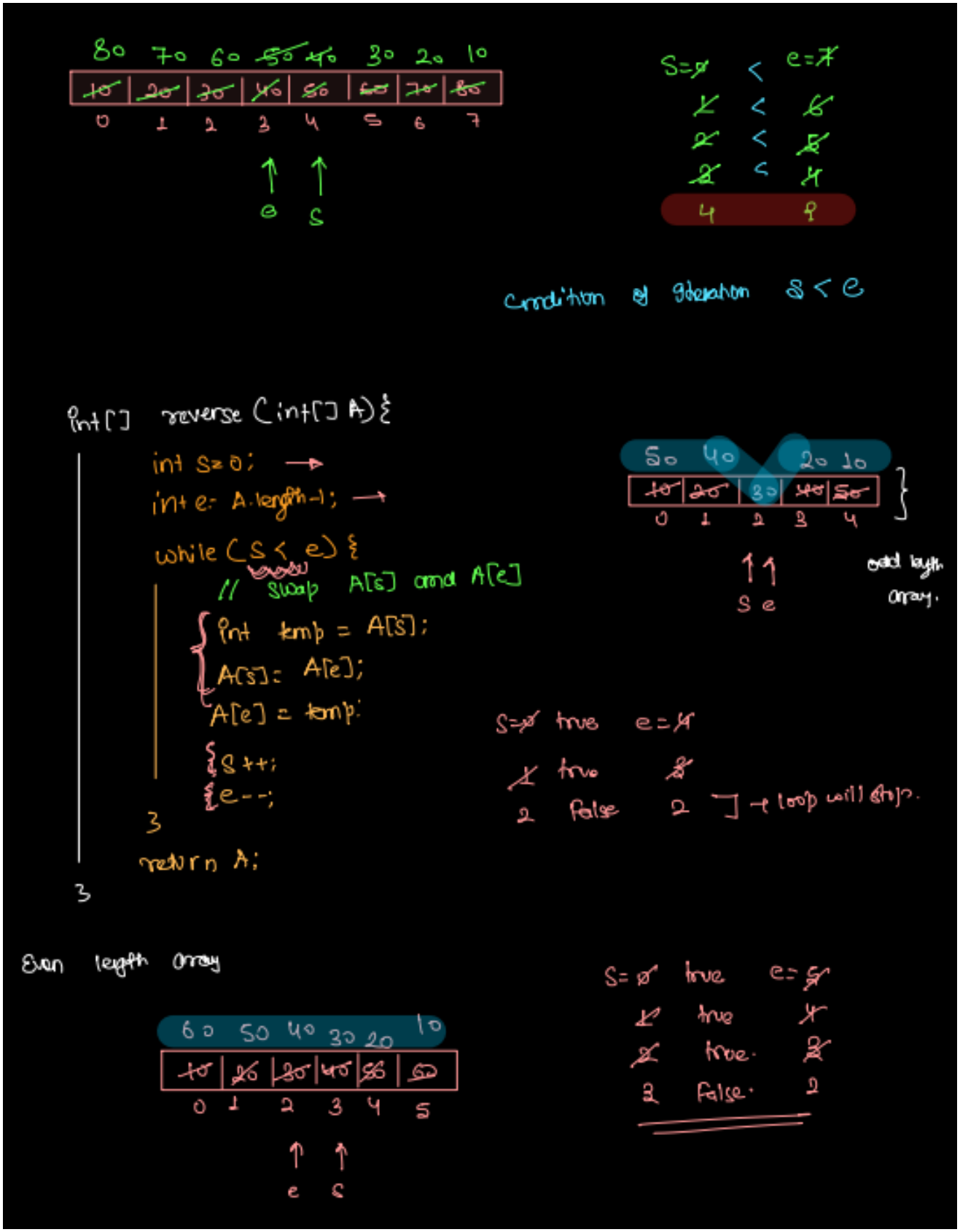
s e

s = 0 → 1 → 2
e = 4 → 3 → 2

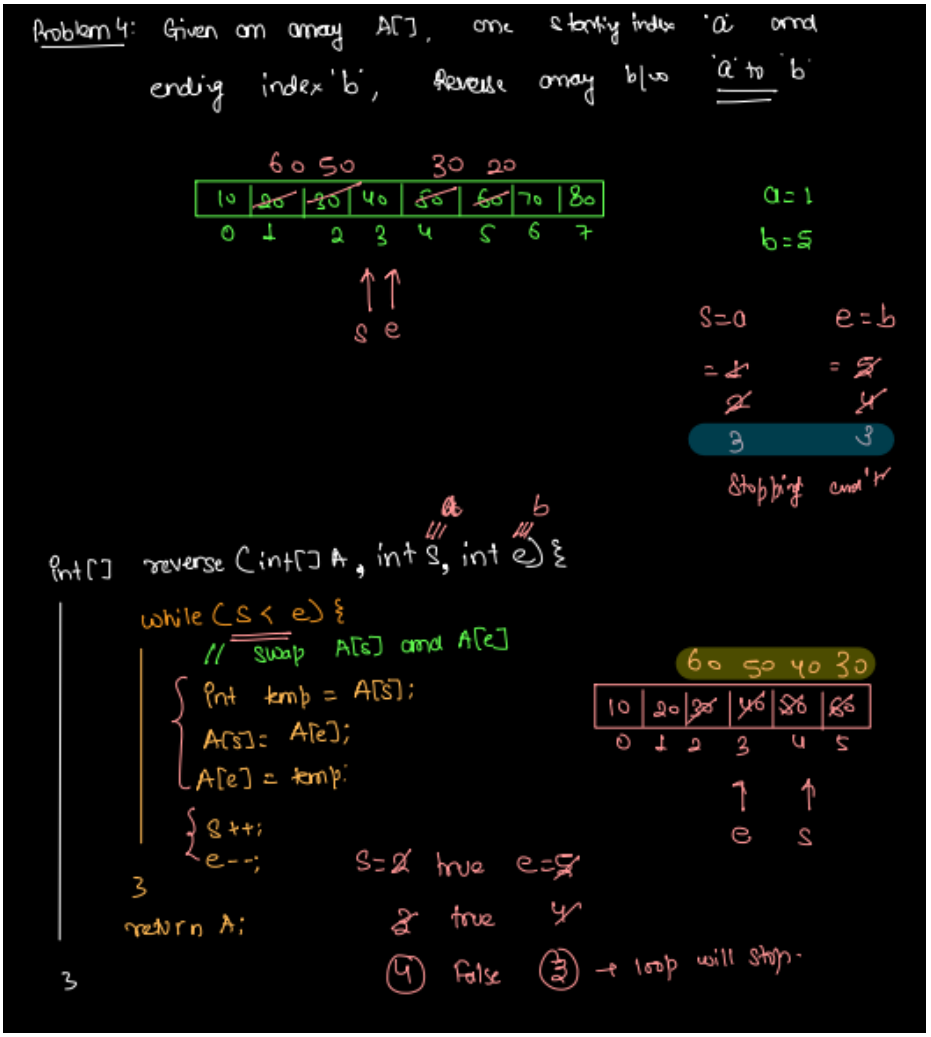
Stopping condition.

Condition of Iteration

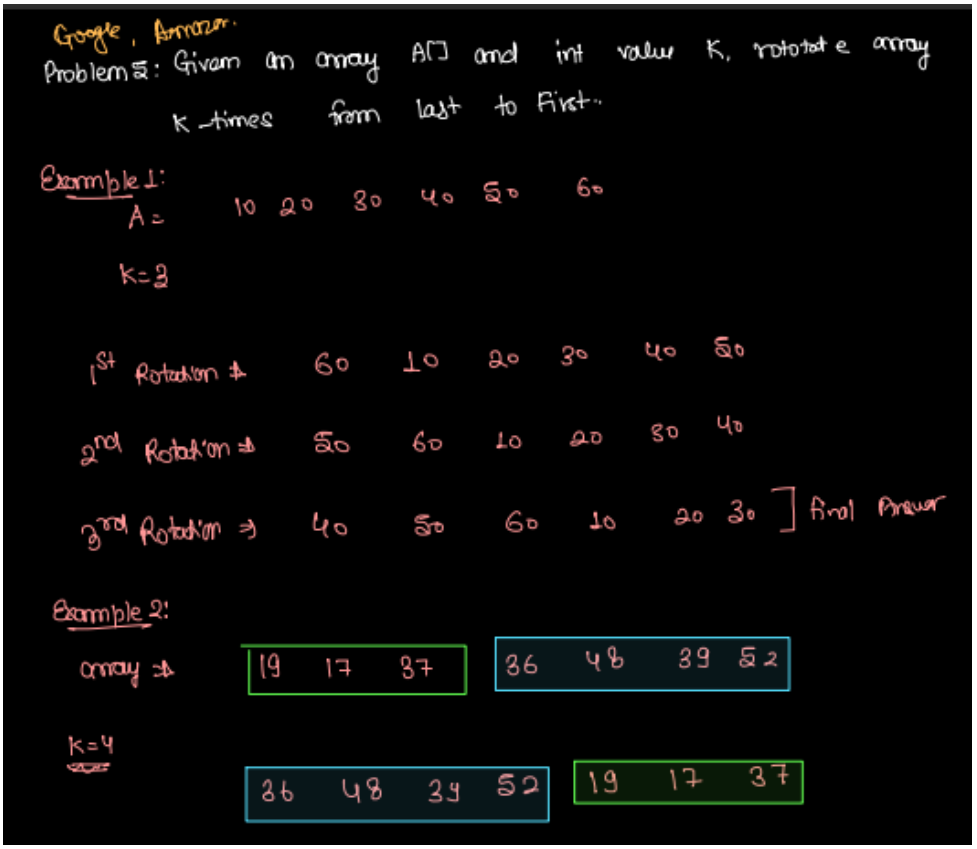
s < e

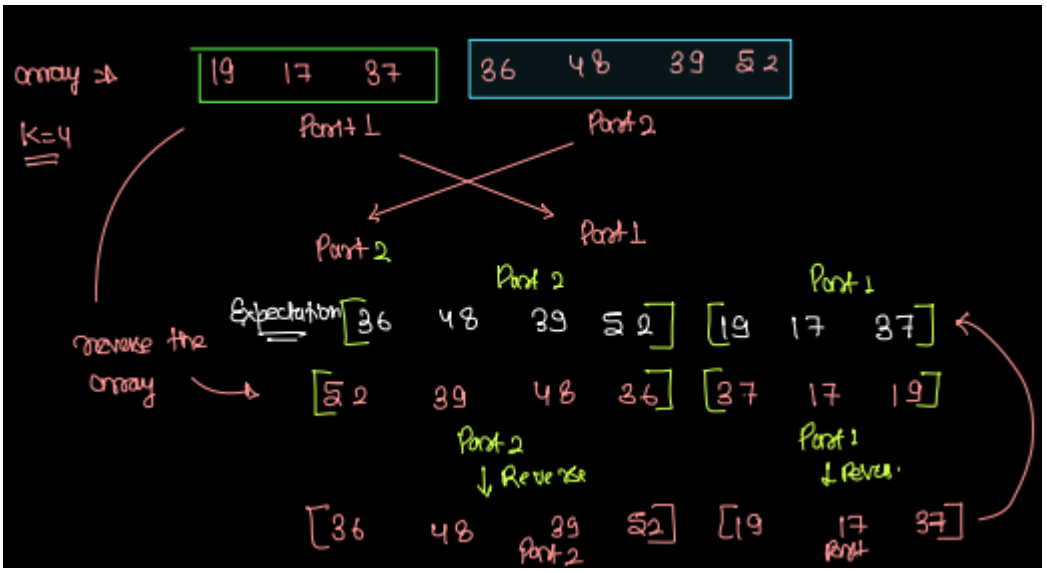


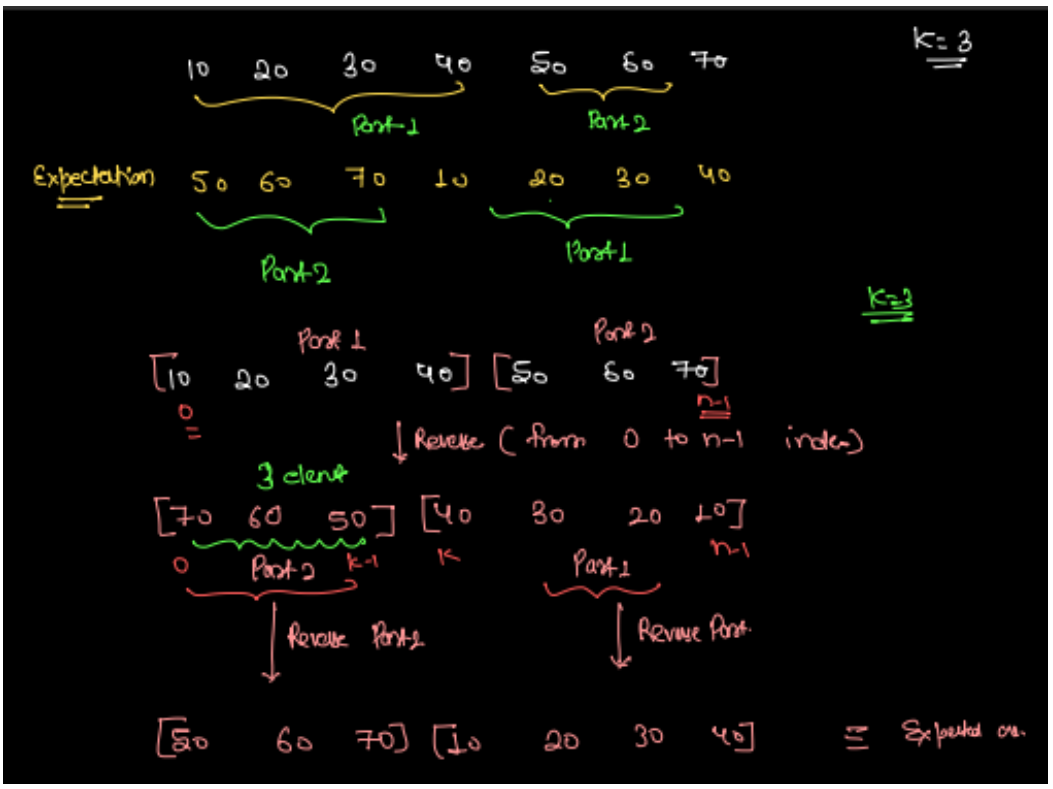
- **Reverse part of an array**

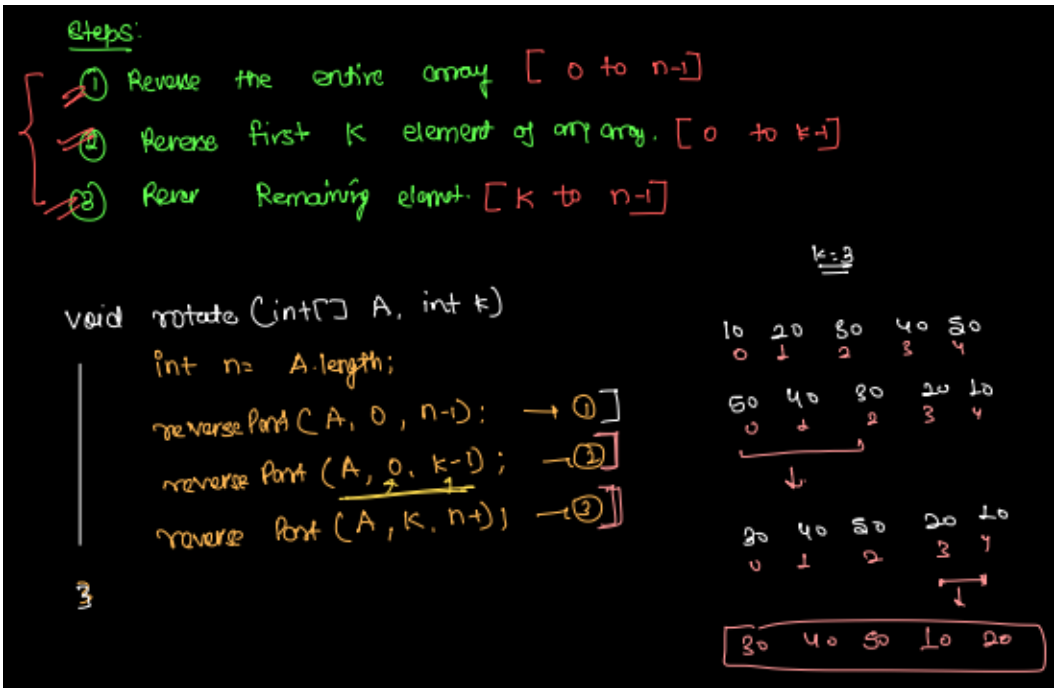


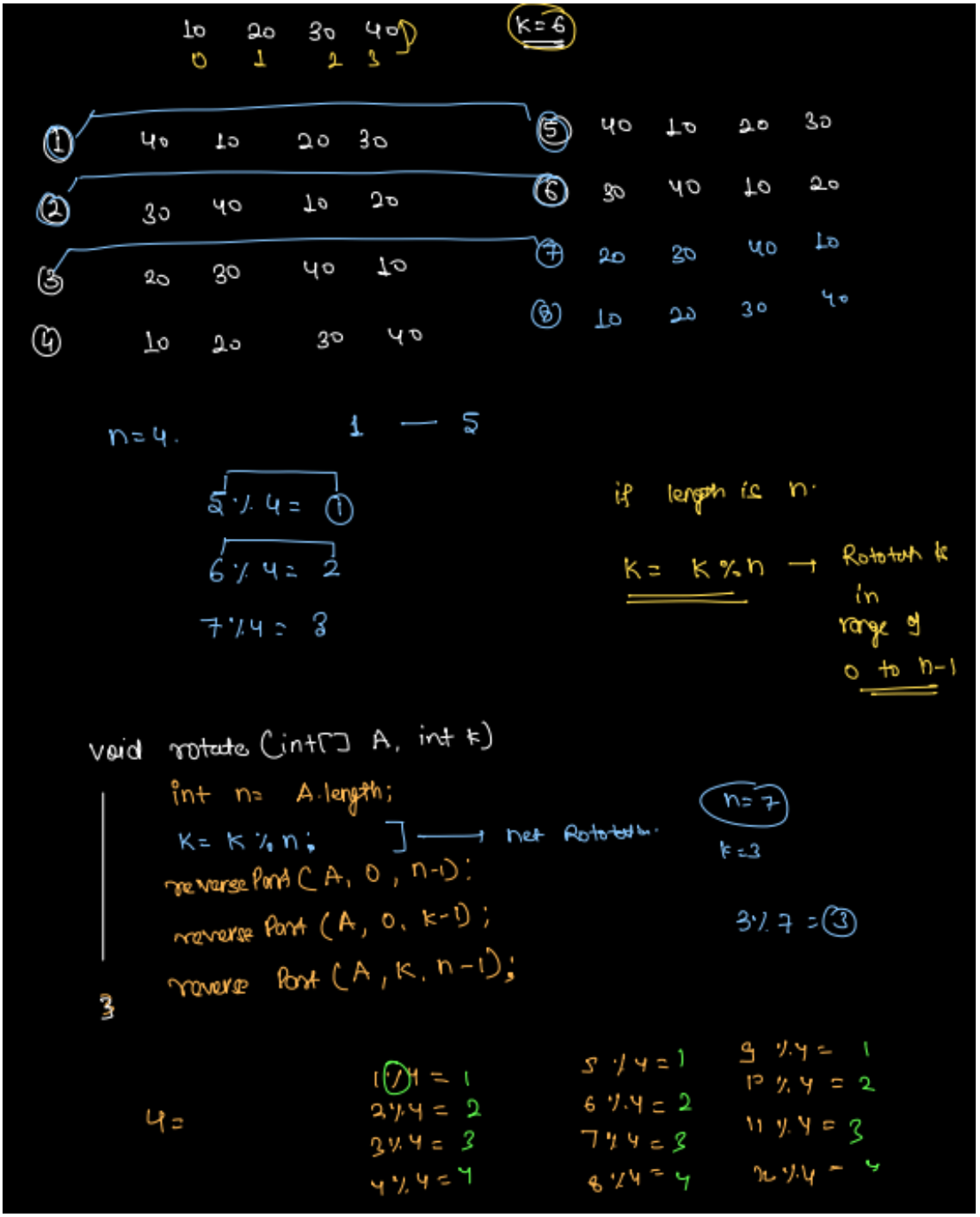
- **Rotate an array**



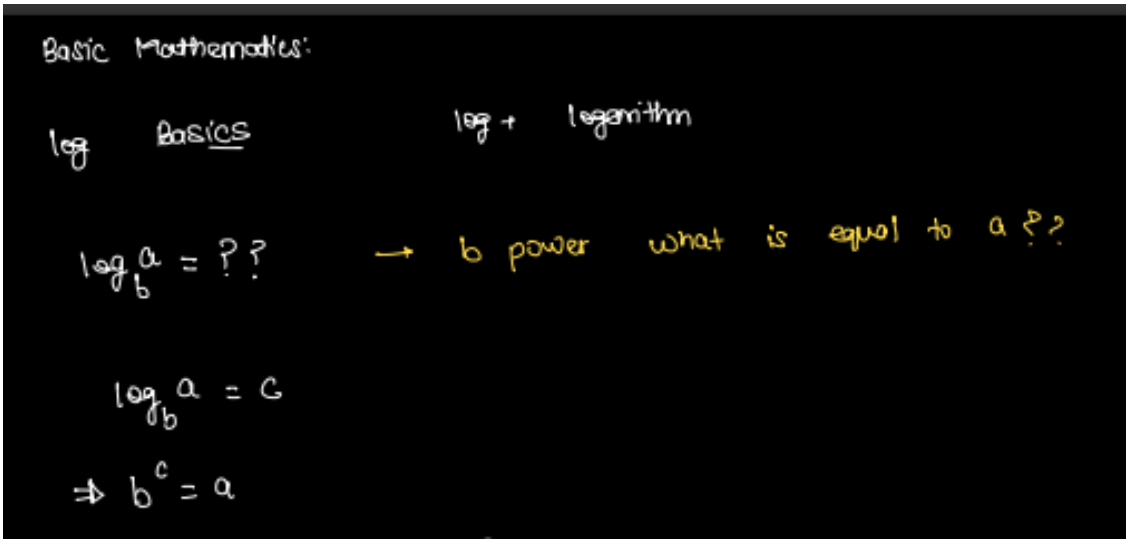

array =
19 17 37
36 48 39 52
K=4
Part 1
Part 2
Part 2
Part 1
Expectation
36 48 39 52 19 17 37
reverse the array
52 39 48 36 37 17 19
Part 2 ↓ Reverse
Part 1 ↓ Reverse
36 48 39 52 19 17 37


K=3
10 20 30 40 50 60 70
Part-1
Part-2
Expectation
50 60 70 10 20 30 40
Part-2
Part-1
K=3
Part 1
Part 2
10 20 30 40 50 60 70
0
n-1
Reverse (from 0 to n-1 index)
3 elements
70 60 50 40 30 20 10
0 Part-2 k-1
K
Part-1
n-1
Reverse Part-2
Reverse Part-1
50 60 70 10 20 30 40
= Expected ans.


Steps:
1 Reverse the entire array [0 to n-1]
2 Reverse first K element of any array. [0 to k-1]
3 Rever Remaining element. [K to n-1]
void rotate (int[] A, int K)
int n = A.length;
reverse Part (A, 0, n-1); → 1
reverse Part (A, 0, k-1); → 2
reverse Part (A, K, n-1); → 3
3
K=3
10 20 30 40 50
0 1 2 3 4
50 40 30 20 10
0 1 2 3 4
30 40 50 10 20
0 1 2 3 4
30 40 50 10 20


10 20 30 40
0 1 2 3
K=6
1 40 10 20 30
2 30 40 10 20
3 20 30 40 10
4 10 20 30 40
5 40 10 20 30
6 30 40 10 20
7 20 30 40 10
8 10 20 30 40
n=4. 1 — 5
5 % 4 = 1
6 % 4 = 2
7 % 4 = 3
if length is n.
K = K % n → Rotation is in range of 0 to n-1
void rotate (int[] A, int k)
int n = A.length;
K = K % n; → net Rotation.
reverse Part (A, 0, n-1);
reverse Part (A, 0, k-1);
reverse Part (A, k, n-1);
n=7
k=3
3 % 7 = 3
4 =
1 % 4 = 1
2 % 4 = 2
3 % 4 = 3
4 % 4 = 4
5 % 4 = 1
6 % 4 = 2
7 % 4 = 3
8 % 4 = 4
9 % 4 = 1
10 % 4 = 2
11 % 4 = 3
12 % 4 = 4

- **Log Basics**


Basic Mathematics:
log Basics
log → logarithm
$\log_b a$ = ??
→ b power what is equal to a ??
$\log_b a = c$
$\Rightarrow b^c = a$

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

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

$\log_2 8 = ?$ $\quad 2^? = 8$

$\log_2 8 = 3$ $\quad 2^3 = 8$

$\log_{10} 10000 = ?$ $\quad 10^? = 10000$

4 power

$\log_{10} 10000 = 4$

$\log_3 81 = ?$ $\quad 3^? = 81$

$3^? = 3 \times 3 \times 3 \times 3$

$3^? = 3^4$

$\log_3 81 = 4$

$\log_3 27 = x$ $\quad 3^x = 27$

x ??

$\log_3 27 = 3$

$3^x = 3 \times 3 \times 3$

$3^x = 3^3$

$x = 3$

$\log_2 33 = ??$ $\quad 2^? = 33$

$? = 5.xyz$ $\quad 2^5 = 2 \times 2 \times 2 \times 2 \times 2 = 32$

$\log_2 33 = 5.xyz = 5$

Properties:

* $N = a^K$

$\Rightarrow K = \log_2 N$

* $\log_b b^N = N$

$\log_b b^N = ?$

$b^? = b^N \Rightarrow N$

Problem Solving & DSA

- Time Complexity
- Arrays [prefix sum, subarray, sliding window, 2D Matrices] } Data Structure
- Bit Manipulation
- Hashing
- Recursion
- Sorting
- Searching
- 2D Pointer Technique } techniques + Algorithm
- Strings
- LinkedList
- Trees, BST, Heap
- Dynamic Programming
- Graph } D.S.

25 sessions = 4 months

DSA ⇒ Data Structure + Algorithm.

- tree
- graph
- Array
- LinkedList
- Stack
- Queue

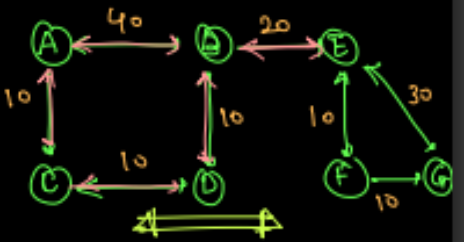
- Kadane's Algo
- dijkstra's
- Krushkal
- bellman ford

Public Transport

A → B → E → F → G

40 20 10 10 ⇒ (80)

A → C → D → B → E → F → G



Shortest dist City1 to City2

A ↔ D distance = (20)

A to G

Graph.

By Bus Road

Shortest distance → City encounter is min

A → B → E → G

A to G

google map

Dijkstra
Krushnac

Hard work → Revision + Dry Run

Problems ⇒ for problems. Scaler is sufficient.

Implement → 8th Q: (20) → 8th Session

Dry run

[Diagonal traverse]

35th Session

DP → Dynamic Programming

Chapter 1

Joy → Struggle ⇔ A

Chapter 21

Game

[25 Q]

DSA: Time Complexity - 1 - 20 - Apr 2023

- **Basic Maths**
- **Iteration Counts in quiz**
- **TC, Comparison and Big-O notation**

- **Sum of natural number ->**

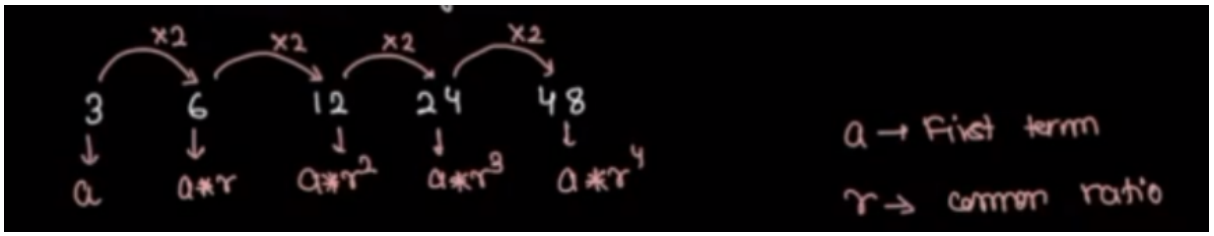
Natural Numbers The numbers that we use when we are counting or ordering {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 …} Whole Numbers The numbers that include natural numbers and zero. Not a fraction or decimal.
sn = N (N+1) / 2

- Inclusive bracket, Exclusive bracket

- **GD (Geometric Progression) ->**

  A geometric progression is a sequence where each term(number) is found by multiplying the previous term(number) by the same number every time. This number is called the **common ratio**.

  like 3, 6, 12, 24, 48



  Common Ratio

  a * r / a = r (ratio) here **a** is first number and **r** is second number in list

  r / a = common ratio - get common ratio

  3 * 6 / 3 = 6 is ratio and 6/3 = 2 is common ratio

  Sum of N(natural numbers) term in the GP (Geometric progression)

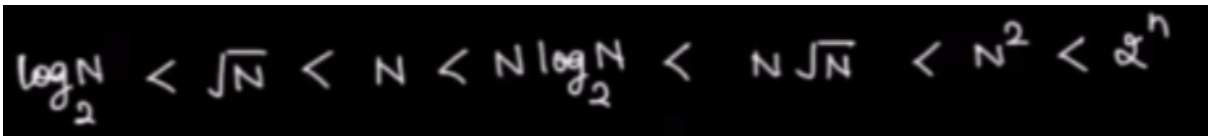
  $S_n = a (r^n - 1) / r - 1$; -> a is first term (number) **r** common ration

  Exponential function ->

  The base 'a' is a positive constant number, usually greater than 0 and not equal to 1.

  - Output will come only positive and greater than 1. we can not get any 0 output either in the negative or positive domain (domain is called which comes on top of x or a). Domain can be - infinity to + infinity.

Comparison of Iterations



How to find Big O notation

- Find total number of Iterations
- Discard lower order terms [Keep higher order term only]
- Discard constant coefficient terms

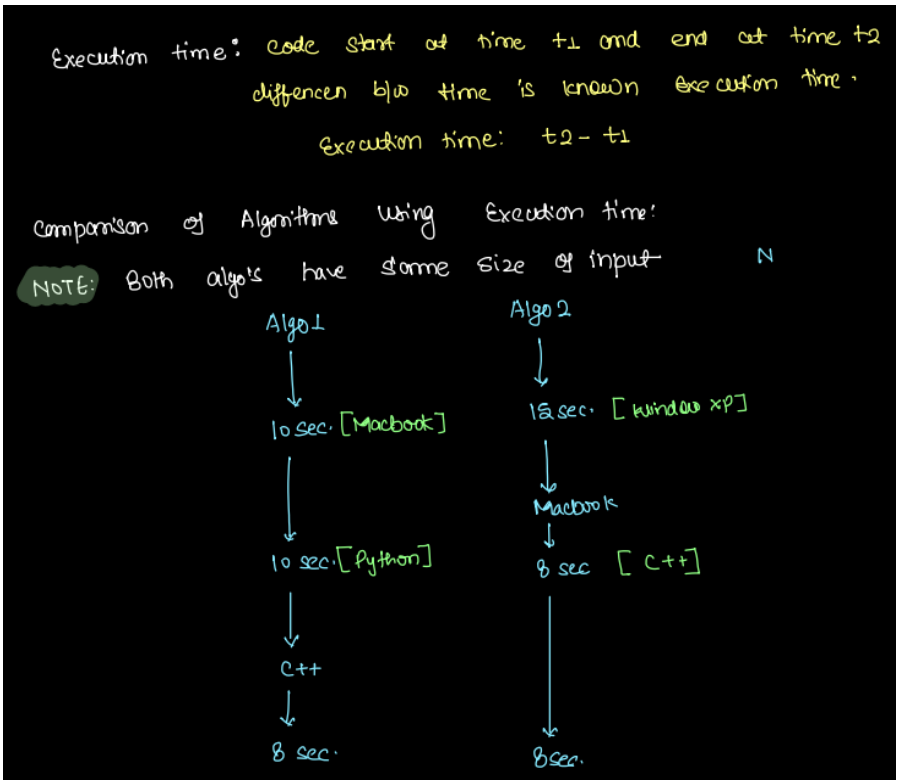
  like $10N^2 + 5N\log N + 6N$ = here N^2 is the highest term

  Time complexity Big O is $O(N^2)$

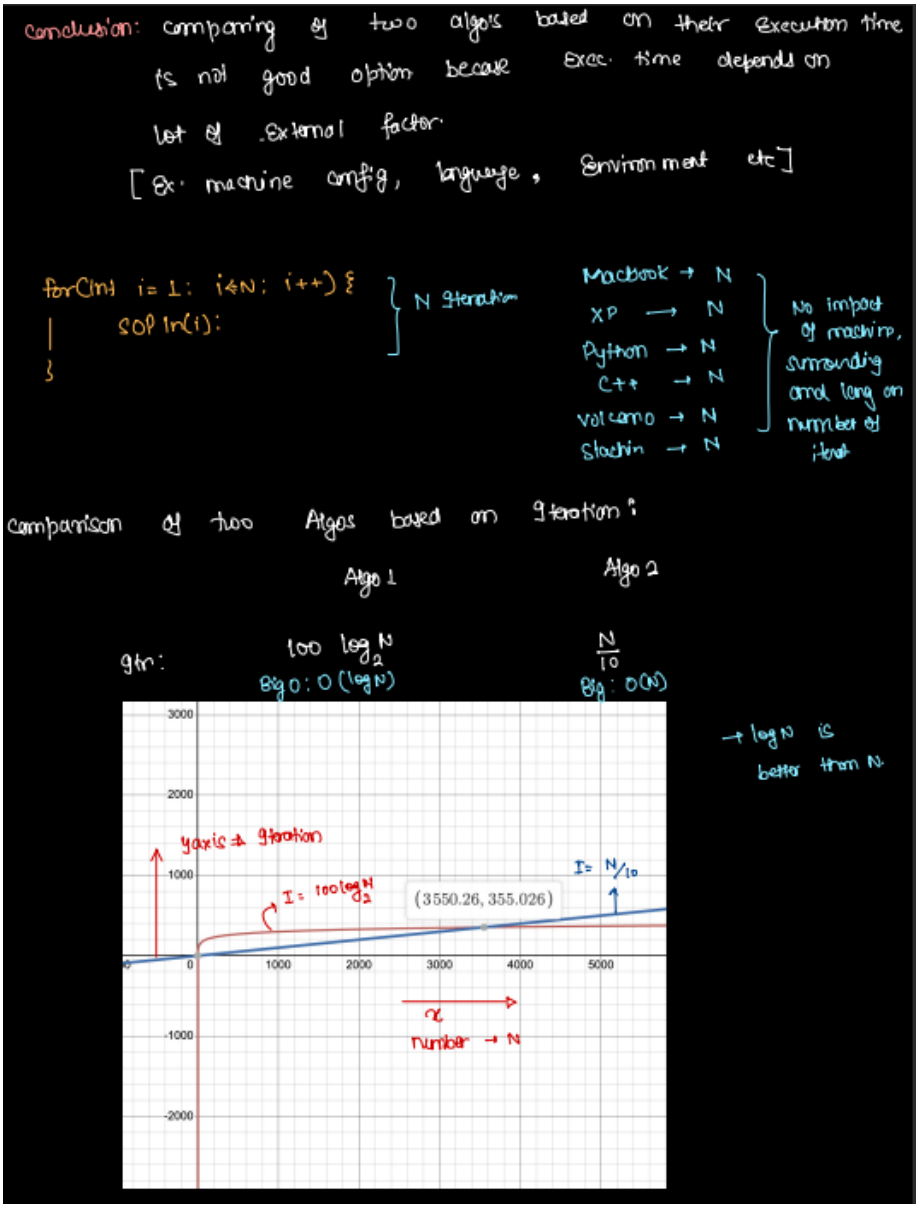
DSA: Time Complexity - 2 - 22 - Apr 2023

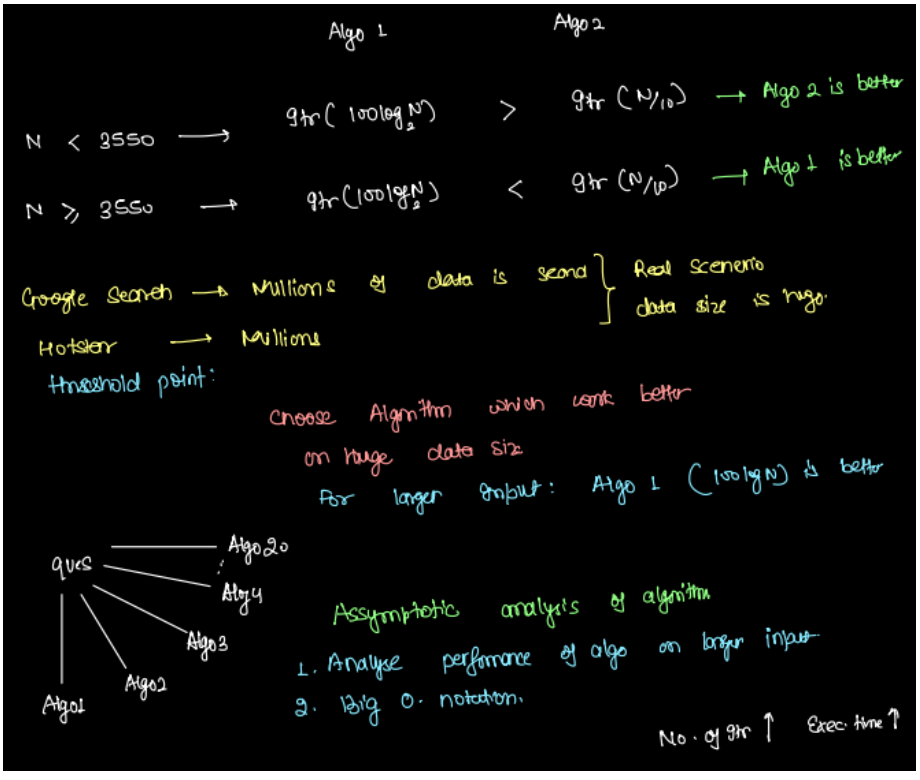
- Comparison Based on execution time
- Comparison based on iteration and graphs
- How to calculate Big-O notation
- Why neglect lower order term?
- Why neglect constant coefficient ?
- issue in big-O notation
- Space complexity
- Time Limit Exceed

- **Comparison Based on execution time**

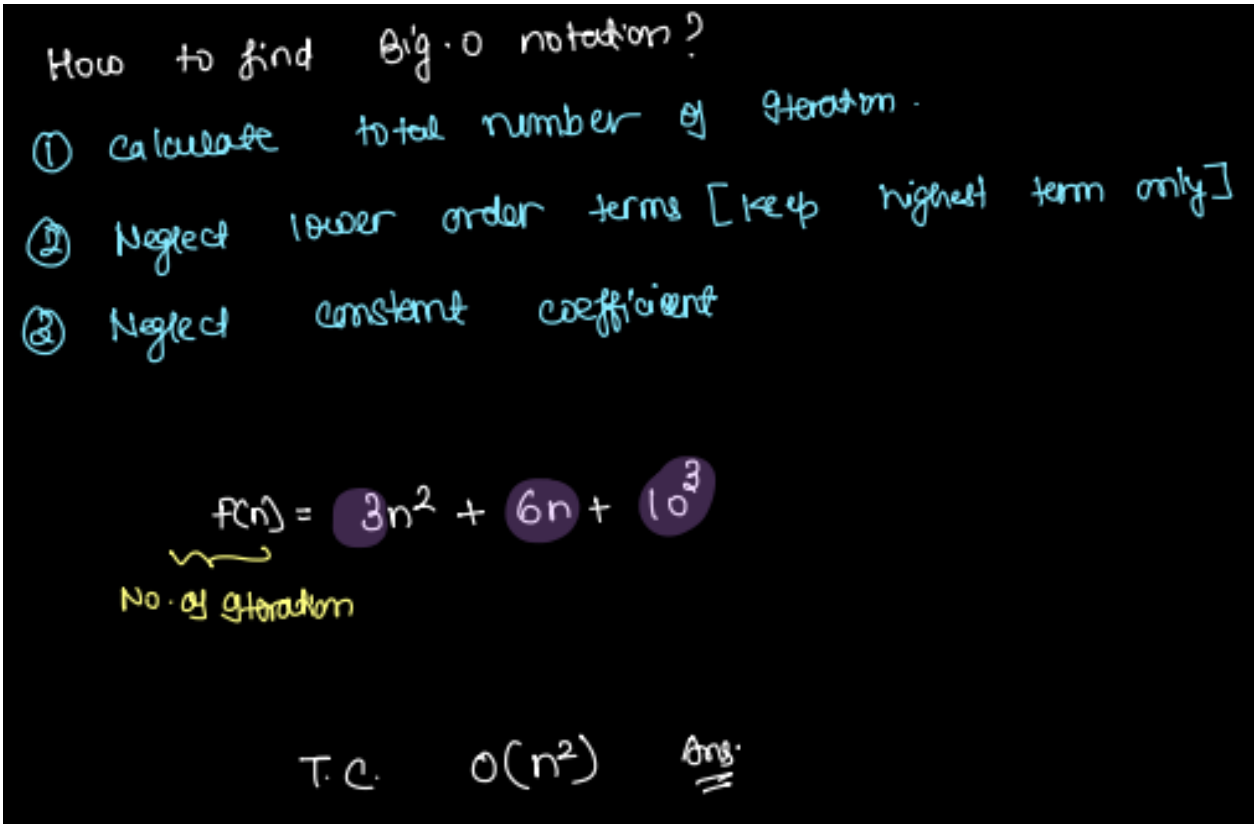


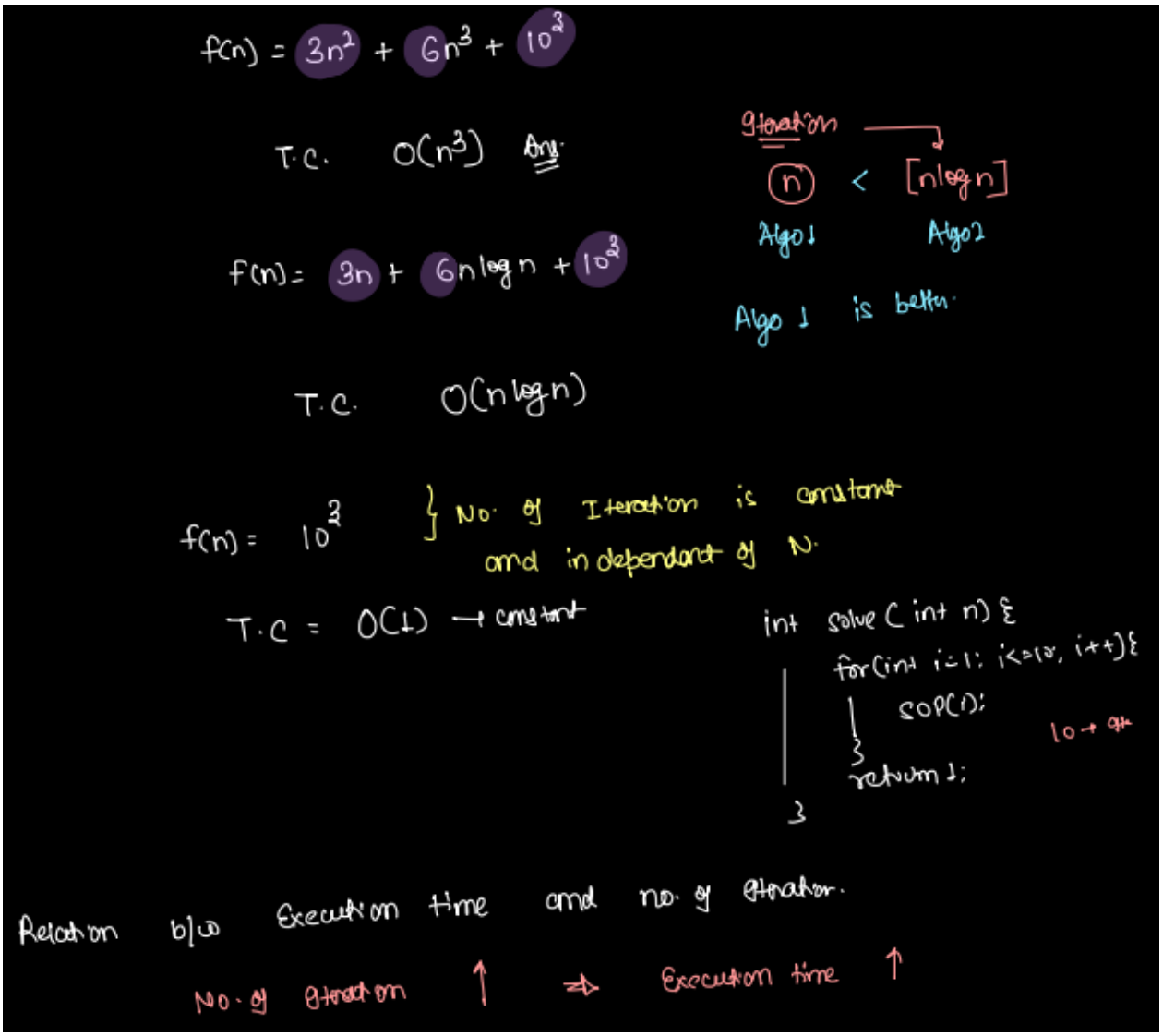
- **Comparison based on iteration and graphs**



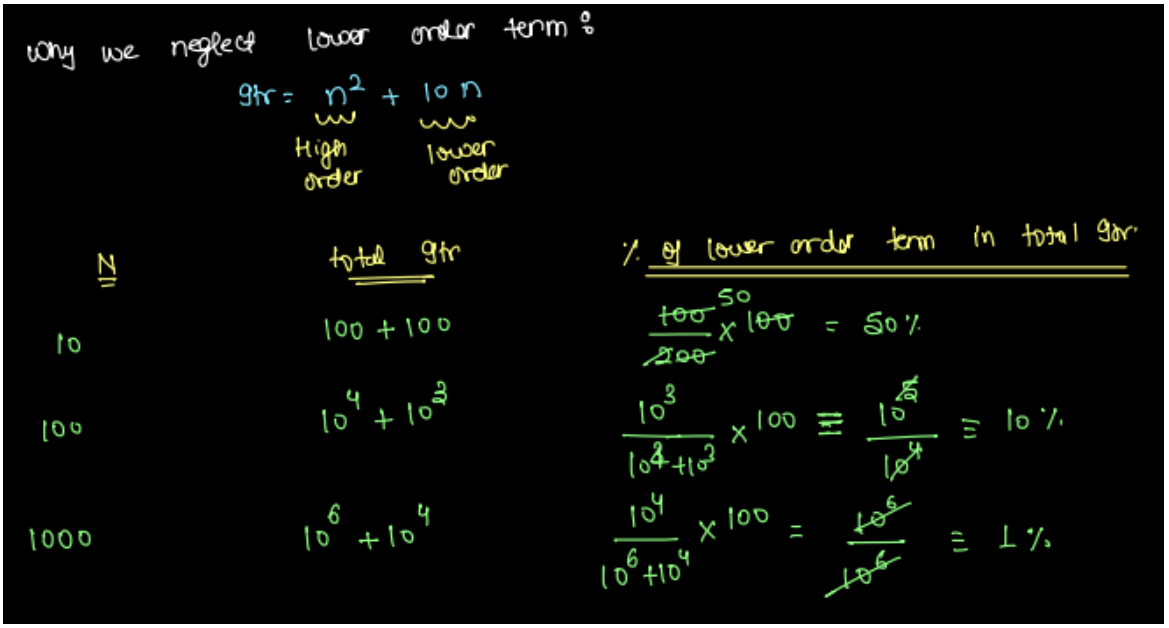


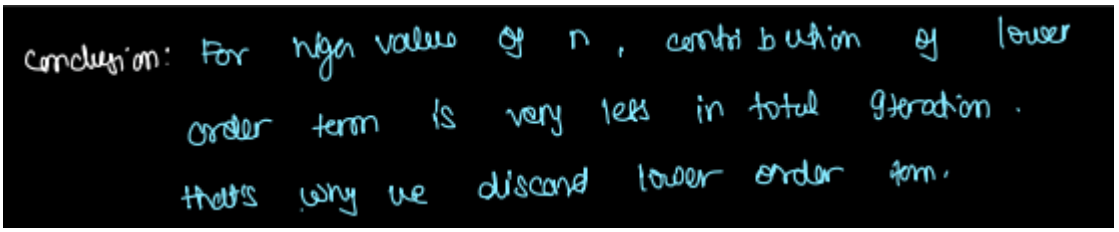
- **How to calculate Big-O notation**



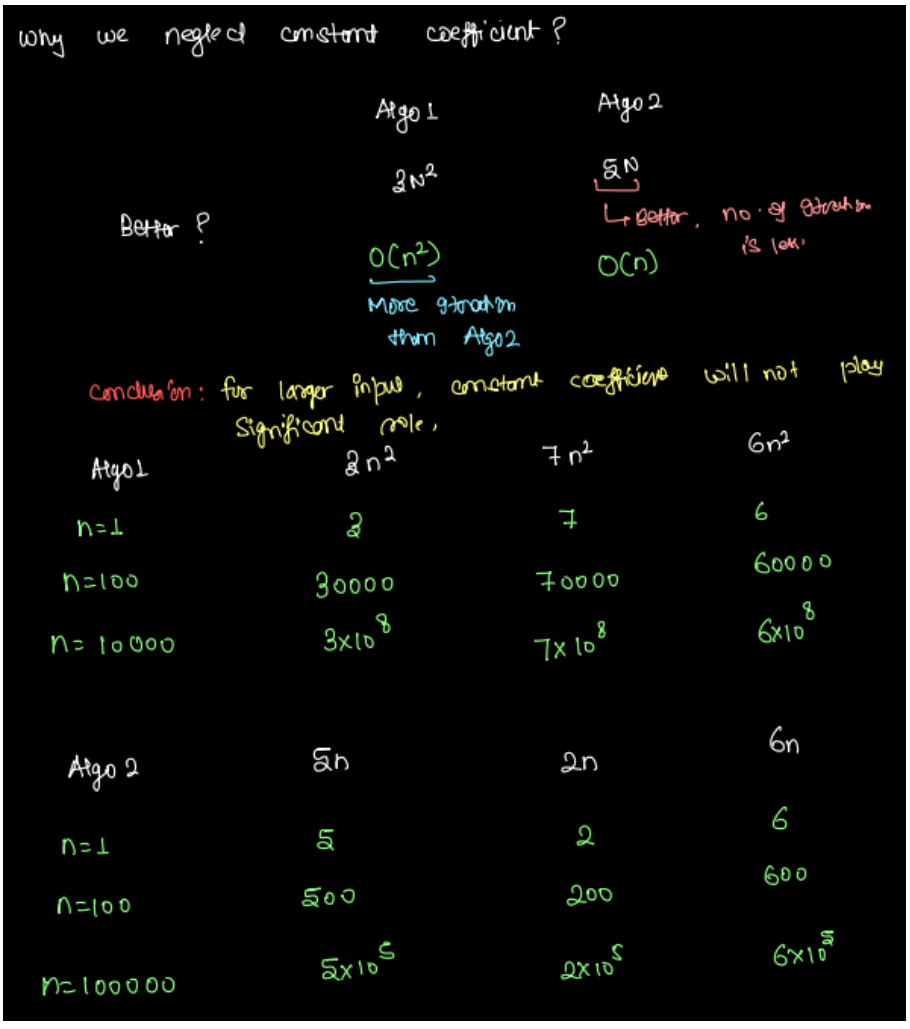


- **Why neglect lower order term?**

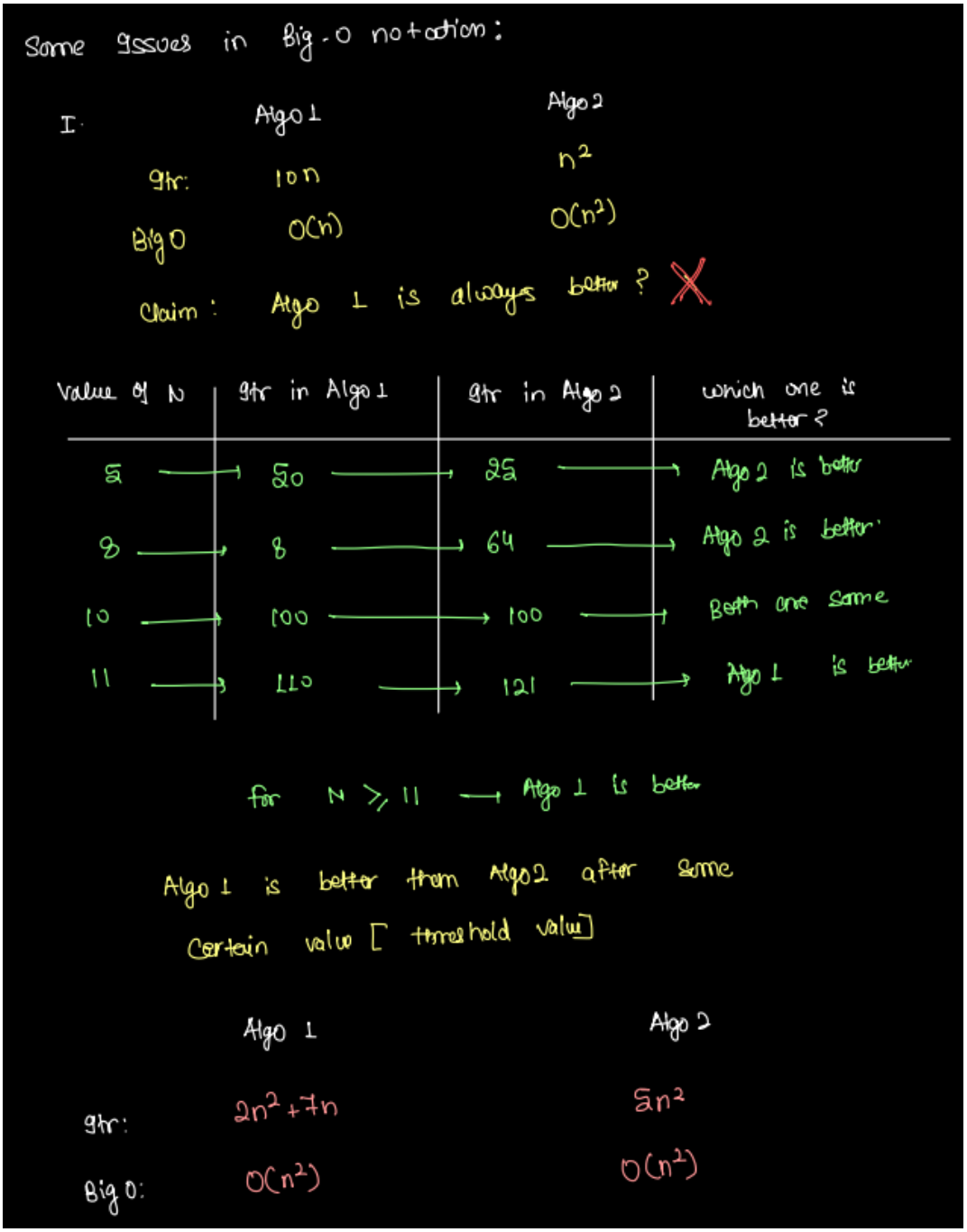


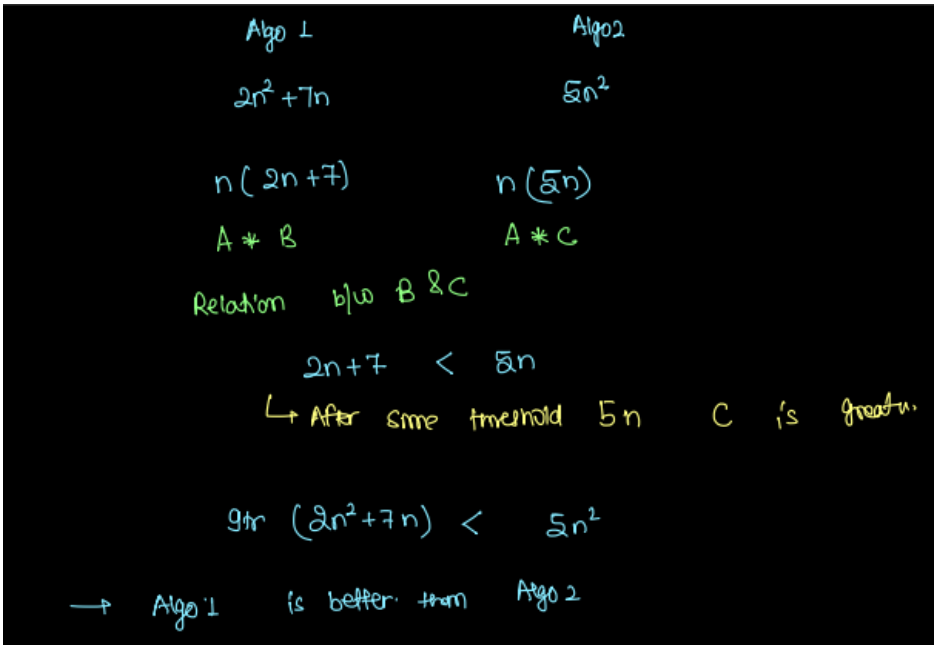


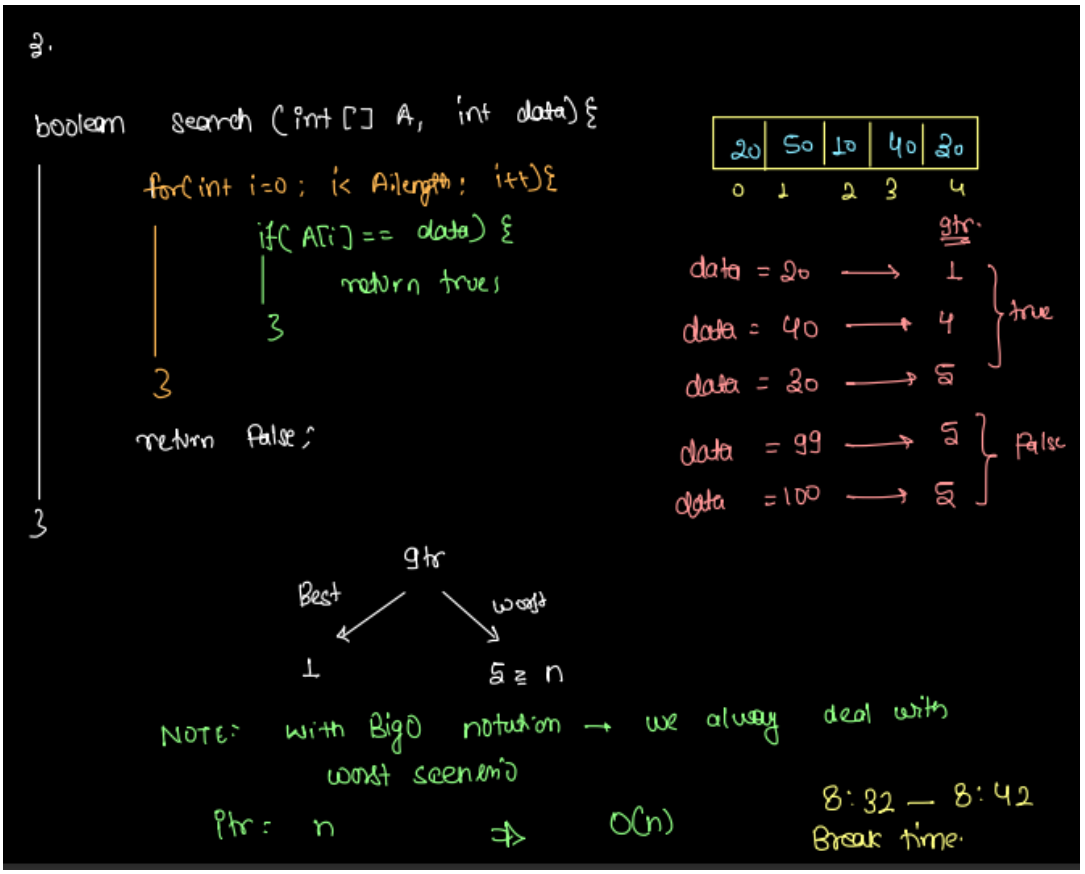
- **Why neglect constant coefficient ?**



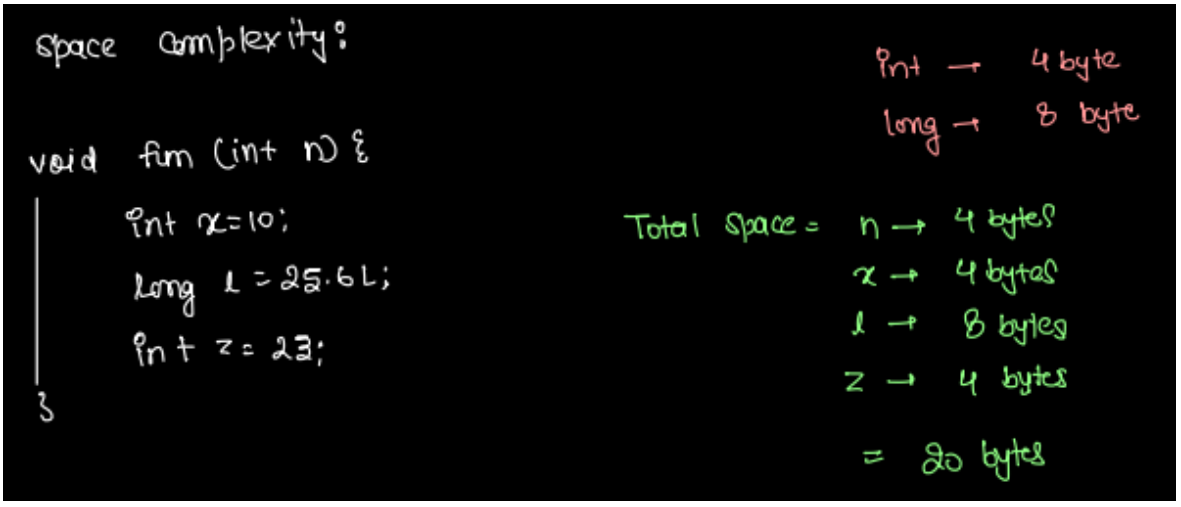
- **issue in big-O notation**

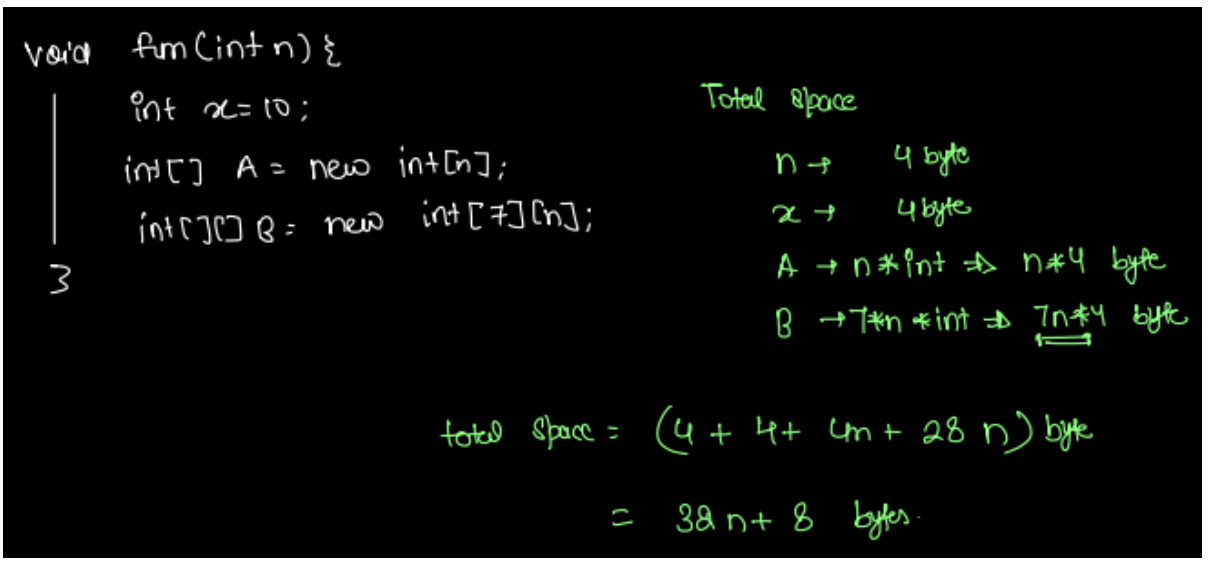






- **Space complexity**





How to calculate space complexity for an Algorithm.

input ⟶ { Algo } ⟶ return o/p.

NOTE: In order to calculate space complexity of an Algo, always consider space which is taken by you. (Don't consider input space in space complexity).

```
int max(int[] A) {
    int mx = A[0];        // 4 byte
    for(int i = 1; i < A.length; i++) {   // 4 byte
        if(A[i] > mx) {
            mx = A[i];
        }
    }
    return mx;
}
```

N → size of array.

T.C.	S.C.
gtr: N	Space complexity
T.C. : O(n)	Space : 8 byte
	S.C = O(1)
	constant space

```
int frequency(int[] A, int k) {
    int count = 0;
    for(int i = 0; i < A.length; i++) {
        if(A[i] == k) {
            count++;
        }
    }
    return count;
}
```

N = A.length

T.C.	S.C.
gtr: n	Space = 8 byte
T.C. : O(n)	S.C. : O(1)

NOTE: Steps for Big-O notation & Space complexity is also same.

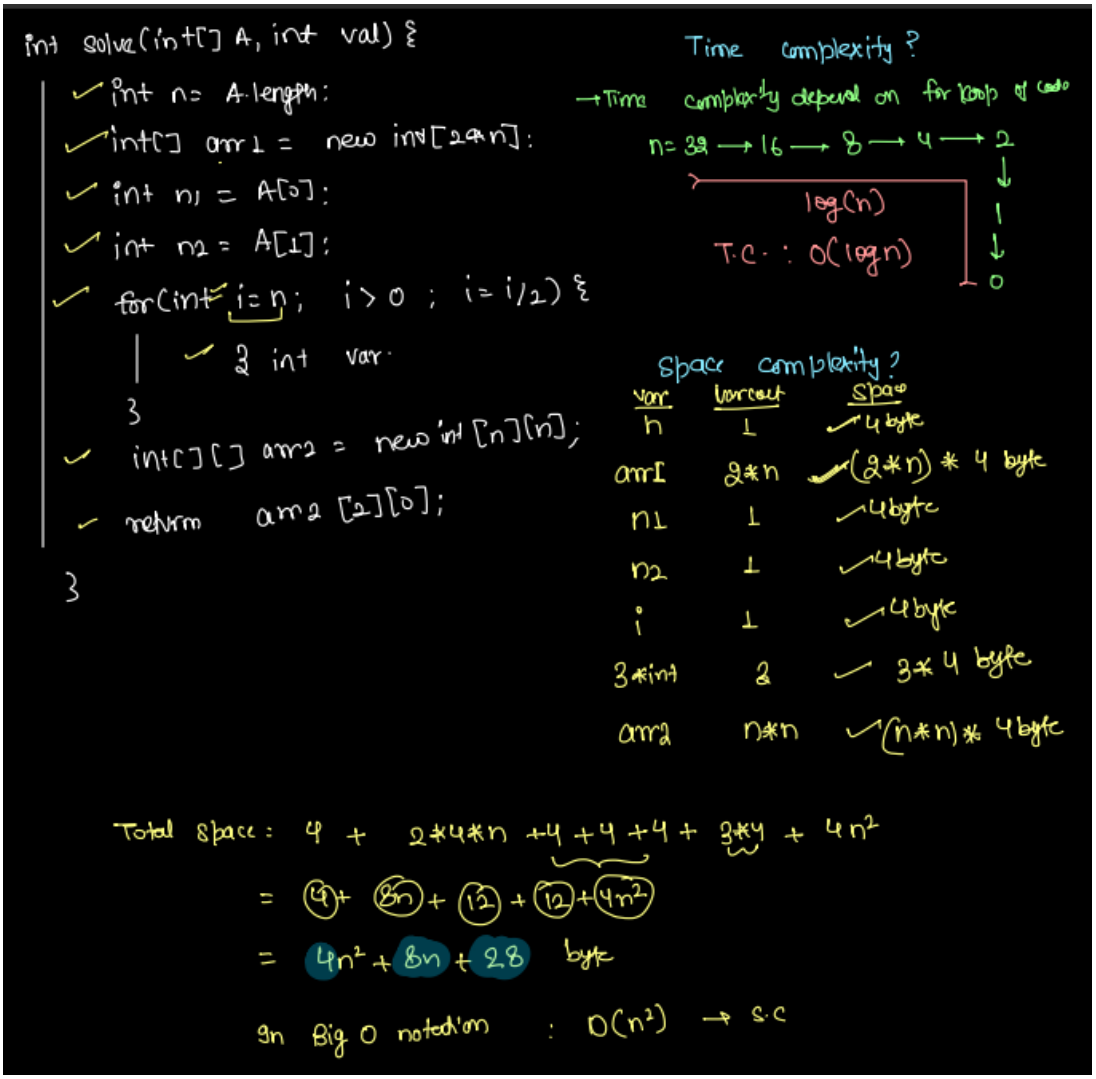
```
int solve(int[] A, int k) {
    int n = A.length;
    int[] B = new int[n];
    for(0 → n times →) {
        [2 int variable
    }
    return B[k];
}
```

T.C.	S.C.
gtr: n	Space
T.C. : O(n)	n → int = 4byte
	B → n*int = n*4byte
	2 int in loop = 2*int = 2*4byte
	i → int : 4 byte

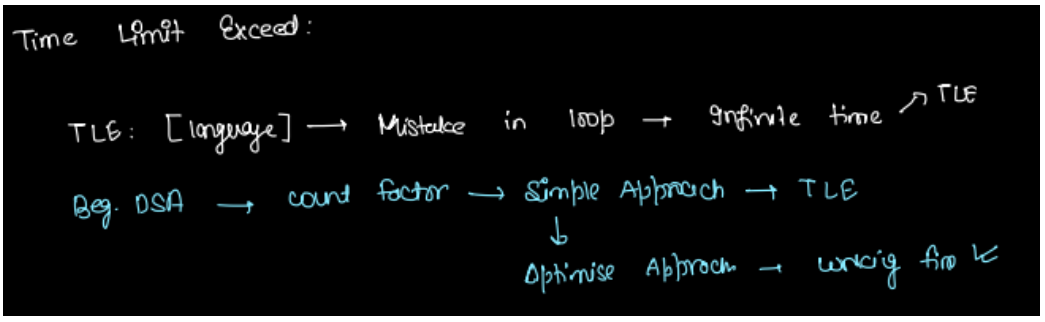
Total Space = 4 + 4n + 8 + 4

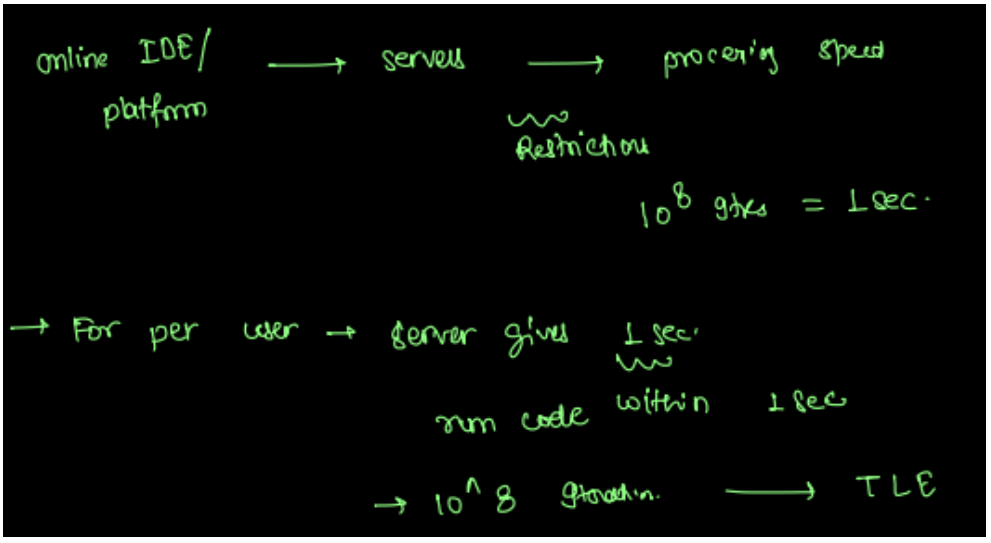
= 4n + 16 bytes.

Big O: O(n)



- **Time Limit Exceed**





Our code should have at max

Safer side → 10^7 to 10^8 ops is fine.

Amazon Contest : 1 hr. 2 problem.

Read → rough code → Code → dry Run → Submit

Submit → TLE → Code

Constraints:

$1 \le N \le 10^5$ max possible value of N = 10^5

Rough code → N^2 ops

if $n = 10^5$

No. of itr = $(10^5)^2 = 10^{10}$

TLE.

↓

don't wait for TLE.

↓

Improve your logic for make iteration $< 10^8$

count factor in Beg. Long.	count factor in Beg. DSA
$1 \le N \le 200$	$1 \le A \le 10^9$
$1 \le N \le 1000$	Brute force, T.C O(n)
Brute force	Max iteration: 10^9
T.C. O(n)	1 sec = 10^8
~	✗ → 10 sec is required.
Max possible iteration 1000	→ TLE with O(n) Approach.
1 sec = 10^8 ops.	optimised Approach
1000 ops will run within 1 sec.	T.C. $O(\sqrt{n})$
	$\sqrt{10^{10}} = 10^5$
	1 sec = 10^8 ops.
	10^5 ops is in range of 1 sec
	→ it will work fine

Problem Constrain

$1 \le n \le 10^{10}$

Algo	T.C	9tr for $n = 10^{10}$
Algo 1 →	n^2 →	10^{20} times → TLE ✗
Algo 2 →	n →	10^{10} times → TLE ✗
Algo 3 →	$\sqrt{n}$ →	$\sqrt{10^{10}} = 10^5$ times ✓
Algo 4 →	$\log_2 n$ →	$10^{10} \equiv 2^{32}$; $\log_2 2^{32} \Rightarrow 32$ times ✓

2 | 100
2 | 50
2 | 25
2 | 12
2 | 6
2 | 3
2 | 1
  | 0

$100 \equiv 2*2*2*2*2*2*2$

$100 \equiv 2^7$

Similarly

$10^{10} \equiv 2^{32}$

How to Approach in interview.

optimised → Reject →] → concept
→ cramming.
→ spoon feed.

$\log n$ → ✓
↓
Brute force
Normal

Brute force
↓
figure out issue
↓
trigger about logic
↓
optimise approach.

Solution → $\log n$
↓
Brute force → n ✗

$100 = 2^7$

$(100)^5 = (2^7)^5$

$10^{10} \equiv 2^{32}$

Equivalent

$35 \equiv 32$

somehow

Approximation

$10^{10} \equiv 2^{35} \equiv 2^{32} \equiv 2^{31}$

$10 \equiv 9$

$10 \equiv 11$

$35 \equiv 32$

What is the time complexity of the following code snippet

```
for(int i = 0 ; i < n ; i++){
    for(int j = 0 ; j <= i ; j++){
        print(i+j);
    }
}
```

i	Range of j	ftr qn I
0	0 to 0	1
1	0 to 1	2
2	0 to 2	3
⋮		⋮
n-1	0 to n-1	n

total ftr = $1 + 2 + 3 + 4 + \dots + n$

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

$= O(n^2)$

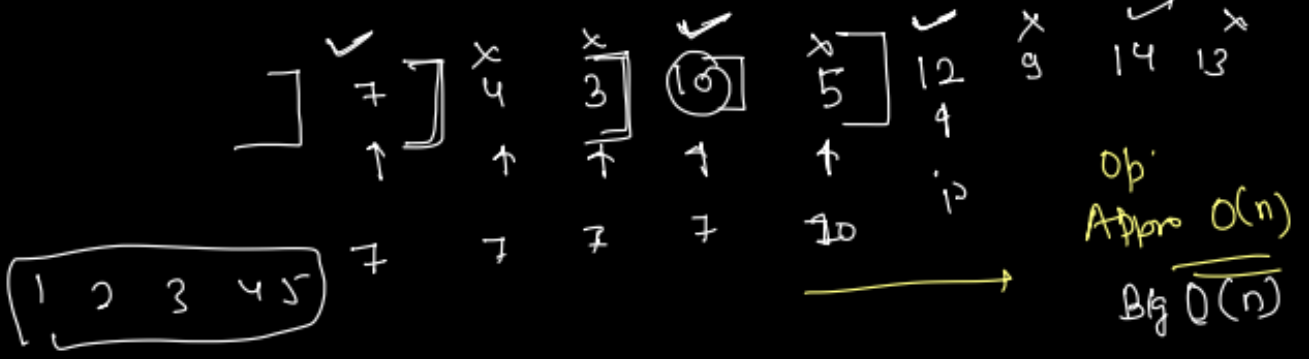
---

leaders in array.

→ [first value is leader] (4)

if A[i] is max value from its left value then it is leader

leader count = ?



1 → 0 to 1 max

2 → 0 to 2 max

3 → 0 to 3 max

4 → 0 to 4 max

⋮

n → 0 to n max

T.C = (n^2)

Big $O(n^2)$

Asymtotic analysis of algorithms -> following point to be noted

- Analysis performance of algorithms on larger input.
- Solve using Big O notation

  How to find big O notation
  - Calculate total number of iterations using logirythem, sum of natural numbers and GP (geometric exponential)
  - Neglect lower order terms [Keep the higher order terms]
  - Neglect and discard constant coefficient

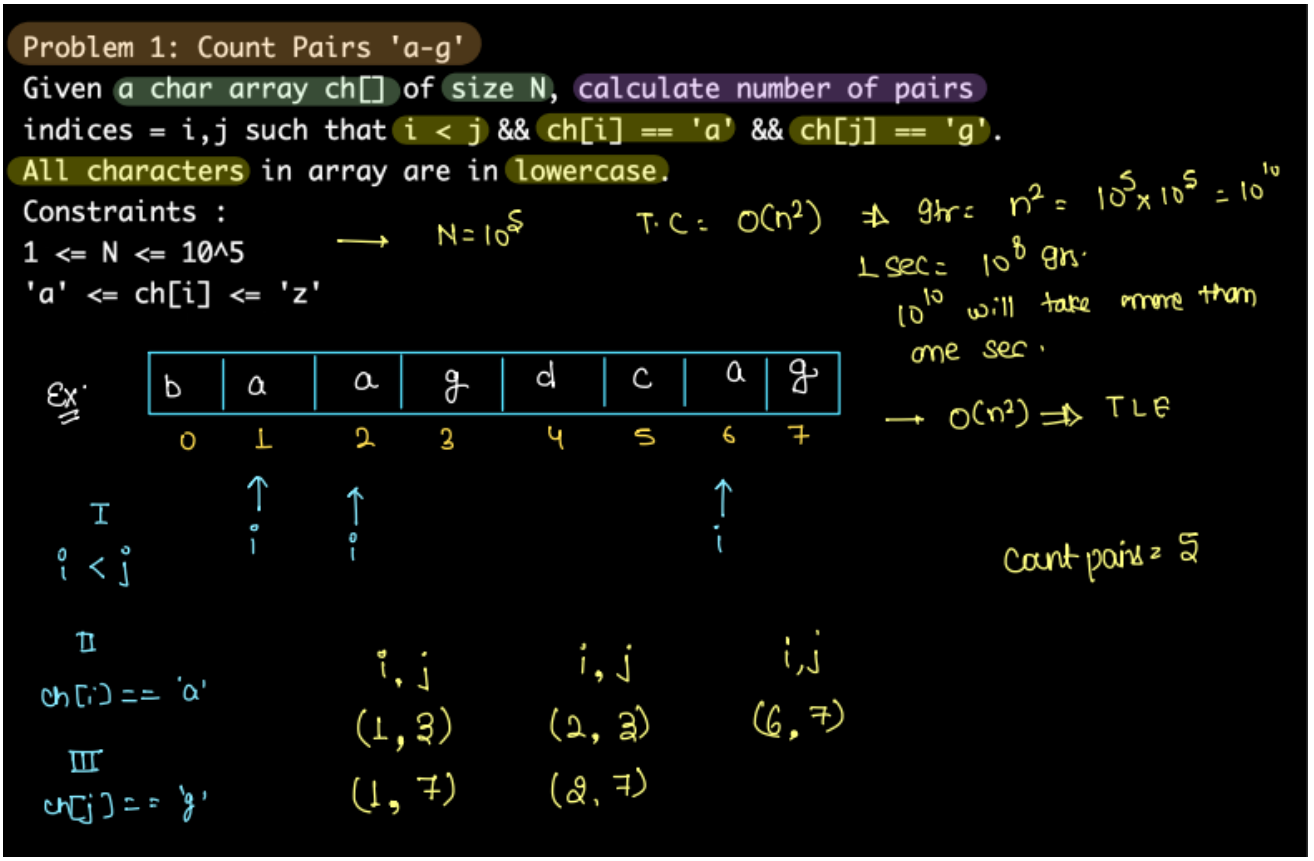
note -> **If No ofIteration is contant and independent of N then time complexity of O(1**)

Why we neglect lower order term because high volume of data(n) contribution of lower order in total iteration is very less

DSA: DSA: Carry Forward - 25 - Apr 2023

- Count Pair a-g
- Leaders in an array
- N Bulbs

- **Count Pair a-g**



Ex 2.

a	c	g	a	g	a	g
0	1	2	3	4	5	6

I
i < j

II
ch[i] == 'a'

III
ch[j] == 'g'

i	j
(0, 2)	(2, 4)
(0, 4)	(3, 6)
(0, 6)	

(i, j)
(5, 6)

total pairs.
6 Ans

Ex.

b	a	g	a	g	g
0	1	2	3	4	5

$a^b * a^c = a^{b+c}$

I
i < j

II
ch[i] == 'a'

III
ch[j] == 'g'

All pairs.

					Pair count
i = 0	(0,1)	(0,2)	(0,3)	(0,4)	(0,5) → 3
i = 1	(1,2)	(1,3)	(1,4)	(1,5)	→ 3
i = 2	(2,3)	(2,4)	(2,5)		→ 2
i = 3	(3,4)	(3,5)			→ 2
i = 4	4,5				→ 1

total pair = (5) pairs.

```
int pairCount(char[] ch) {
    int n = ch.length;
    int count = 0;
    for(int i = 0; i < n; i++) {
        for(int j = i+1; j < n; j++) {
            if( ch[i] == 'a' && ch[j] == 'g') {
                count++;
            }
        }
    }
    return count;
}
```

Value of i	Ranged j	Iteration count in j
0	[1, n)	n-1
1	[2, n)	n-2
2	[3, n)	n-3
3	[4, n)	n-4
⋮		
n-2	[n-1, n)	1
n-1	[n, n)	0

Total Itr = 0+1+2+3+ . . . +n-1
= $\frac{n(n-1)}{2}$

T.C = $O(n^2)$

ch →

a	d	g	a	g	a	g	f	g
0	1	2	3	4	5	6	7	8

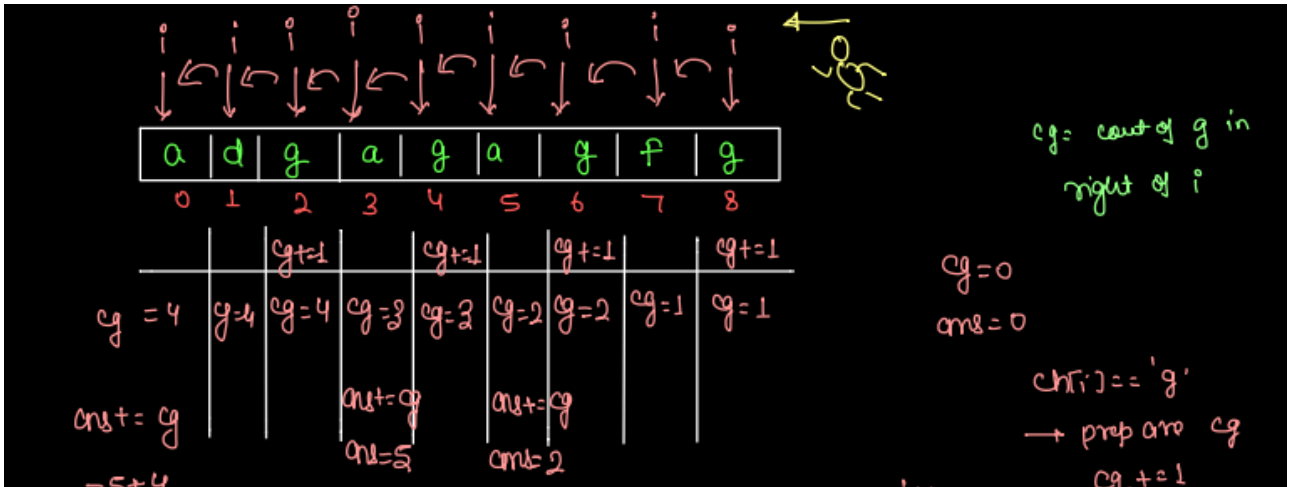
i = 0 → 4 pairs [4 'g's are available in right of i]

i = 2 → 2 pairs

i = 5 → 2 pairs

Total pairs = 4+3+2 = 9 pairs

Optimisation [Right to left]



=5+4

=9

cg = count of g in right of i

cg = 0

ans = 0

ch[i] == 'g' → prep ans cg

cg += 1

ch[i] == 'a' → prep ans

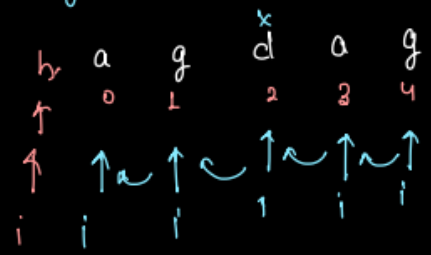
ans += cg.

Ans: 9 pairs

```
int countPairs(char[] ch) {
    int n = ch.length;
    int ans = 0;
    int countg = 0;
    for(int i=n-1; i>=0; i--) {
        if(ch[i] == 'g') {
            countg += 1;
        } else if (ch[i] == 'a') {
            ans += countg;
        }
    }
    return ans;
}
```

ans = ∅ ↙ 2

countg = ∅ ↙ 2



gtr. count = [0 to n-1]

= n

T.C.: O(n)

S.C.: O(1)

Summary: ① Iteration from Right to left

Why?

count of g from i+1 to last index.

Iterate from Right to Left; we can track the count of 'g' from i+1 to n-1.

② if ch[i] == 'g' { countg += 1 }

Increase g count by 1.

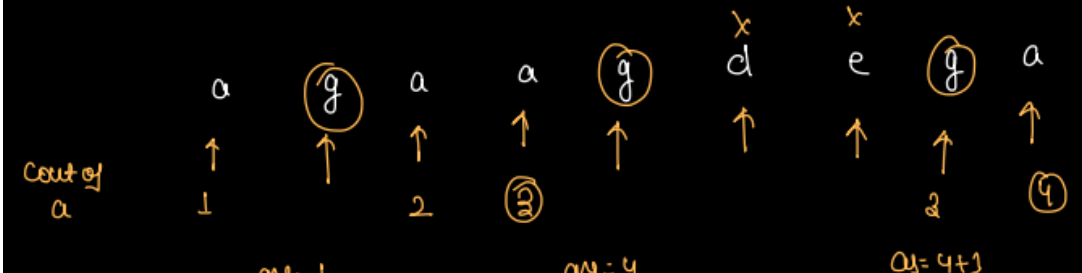
③ if ch[i] == 'a' { ans += count of g }

if char at ith index is 'a' & we have count of g available in right.

Possible pairs are nothing but g count in right side of ith index

→ ans += count of g.

Method 2: [Iteration from left to Right]



TODO:

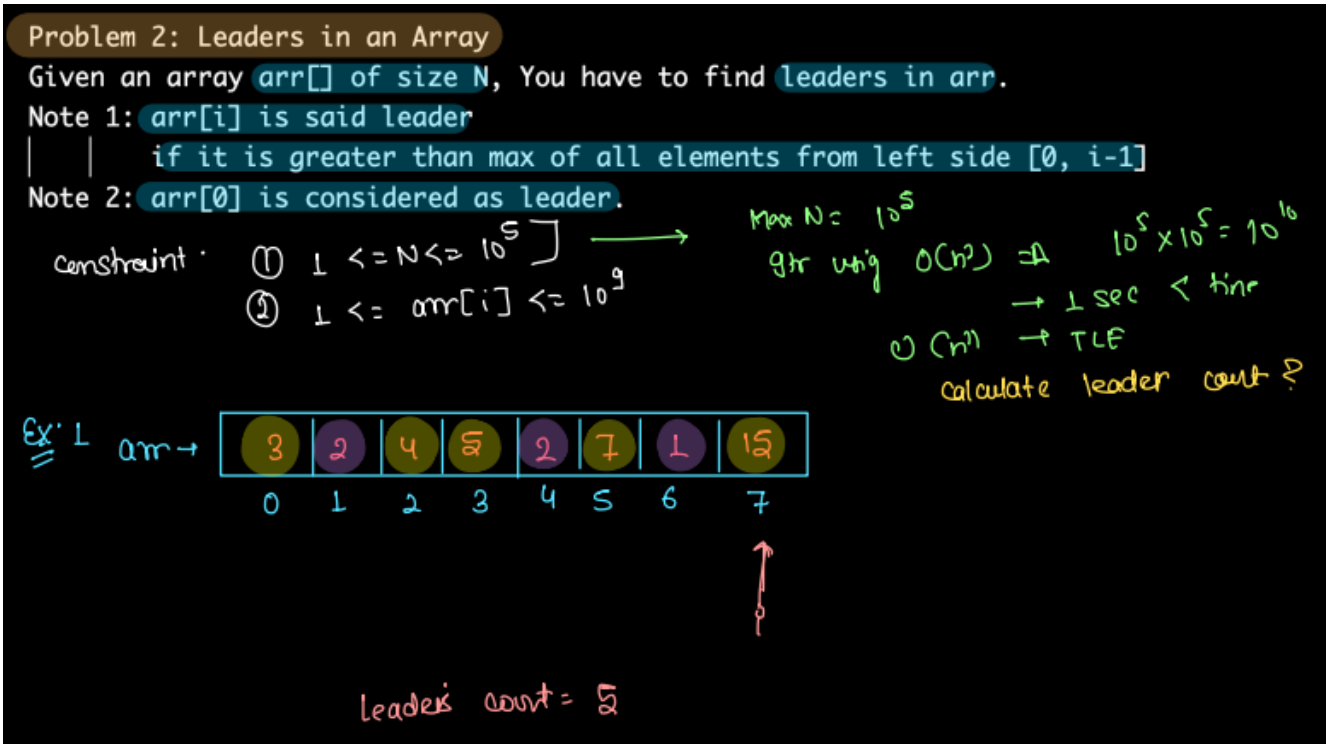
→ Iterate from left to right

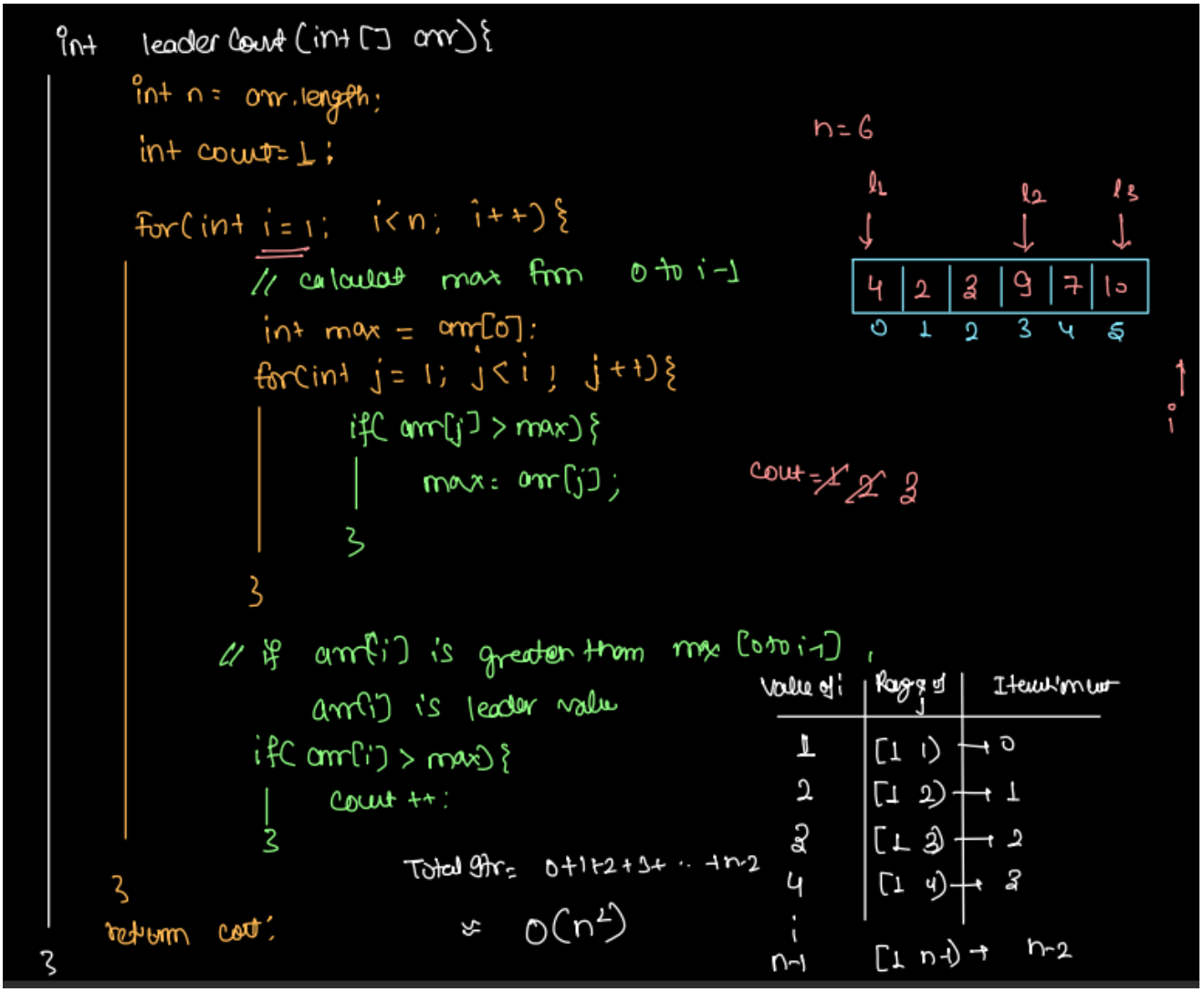
→ manage count of 'a'

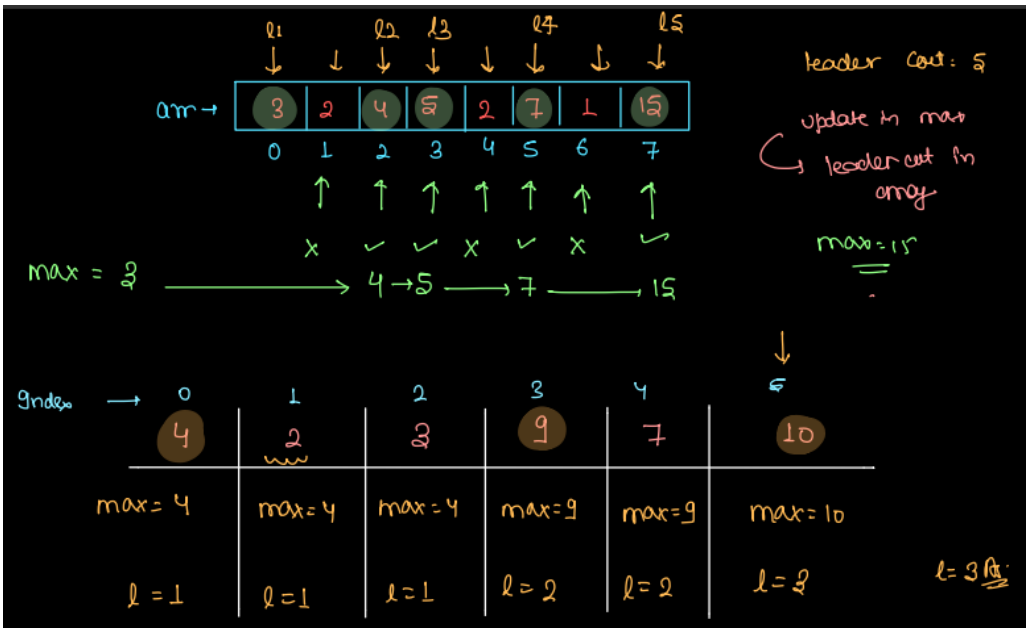
→ if 'g' encounts prepare your ans.

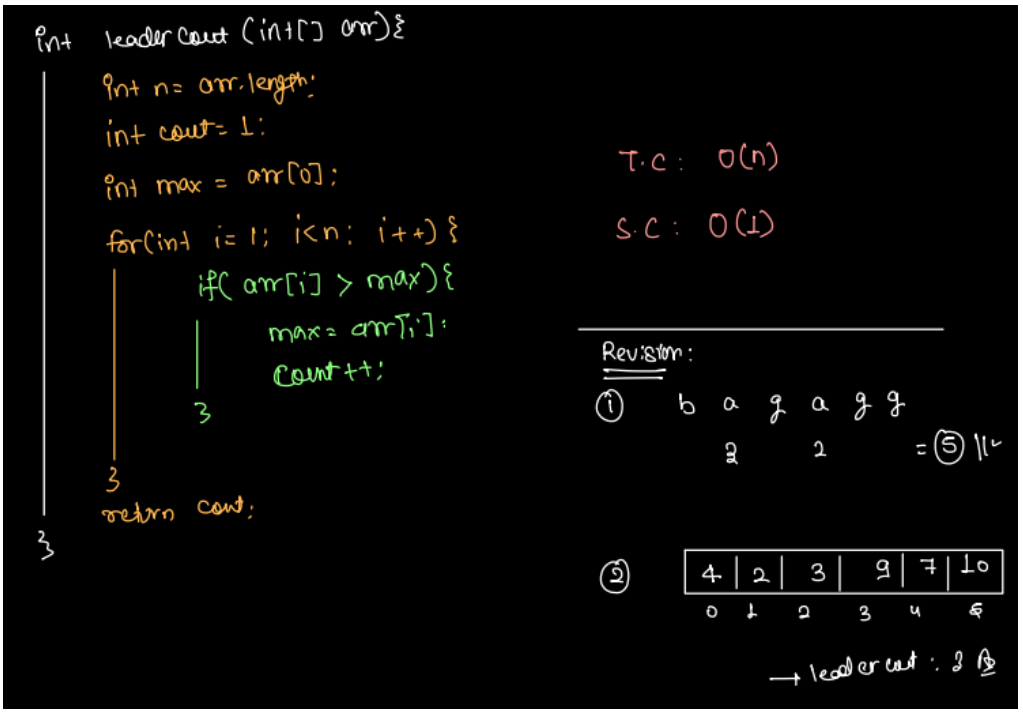
[Break time]: 8:11 – 8:25

- **Leaders in an array**

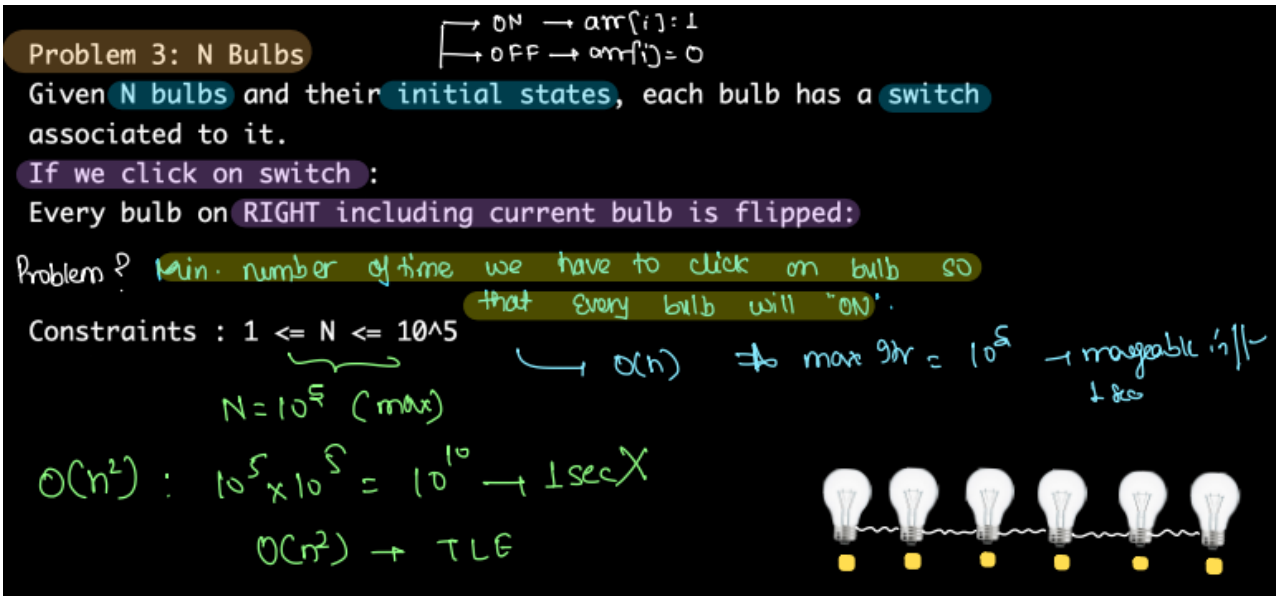


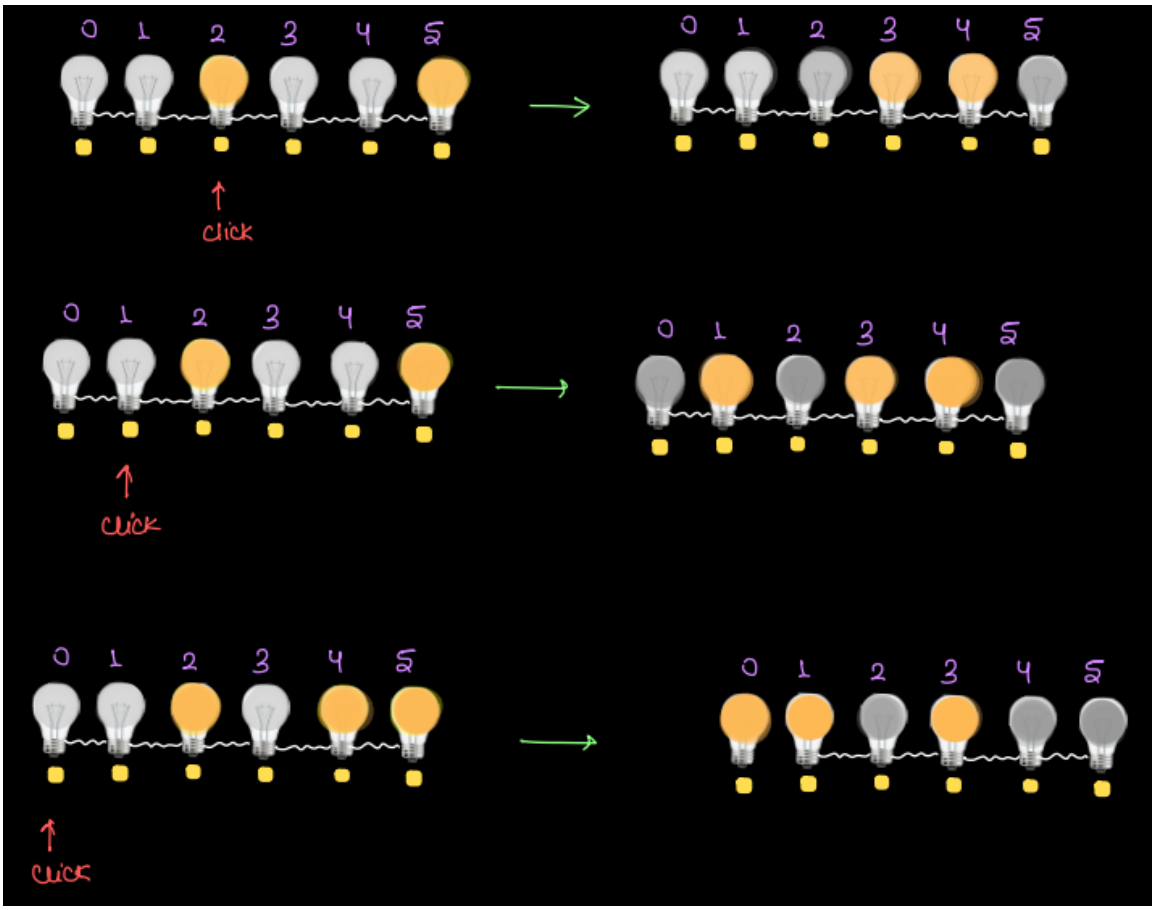


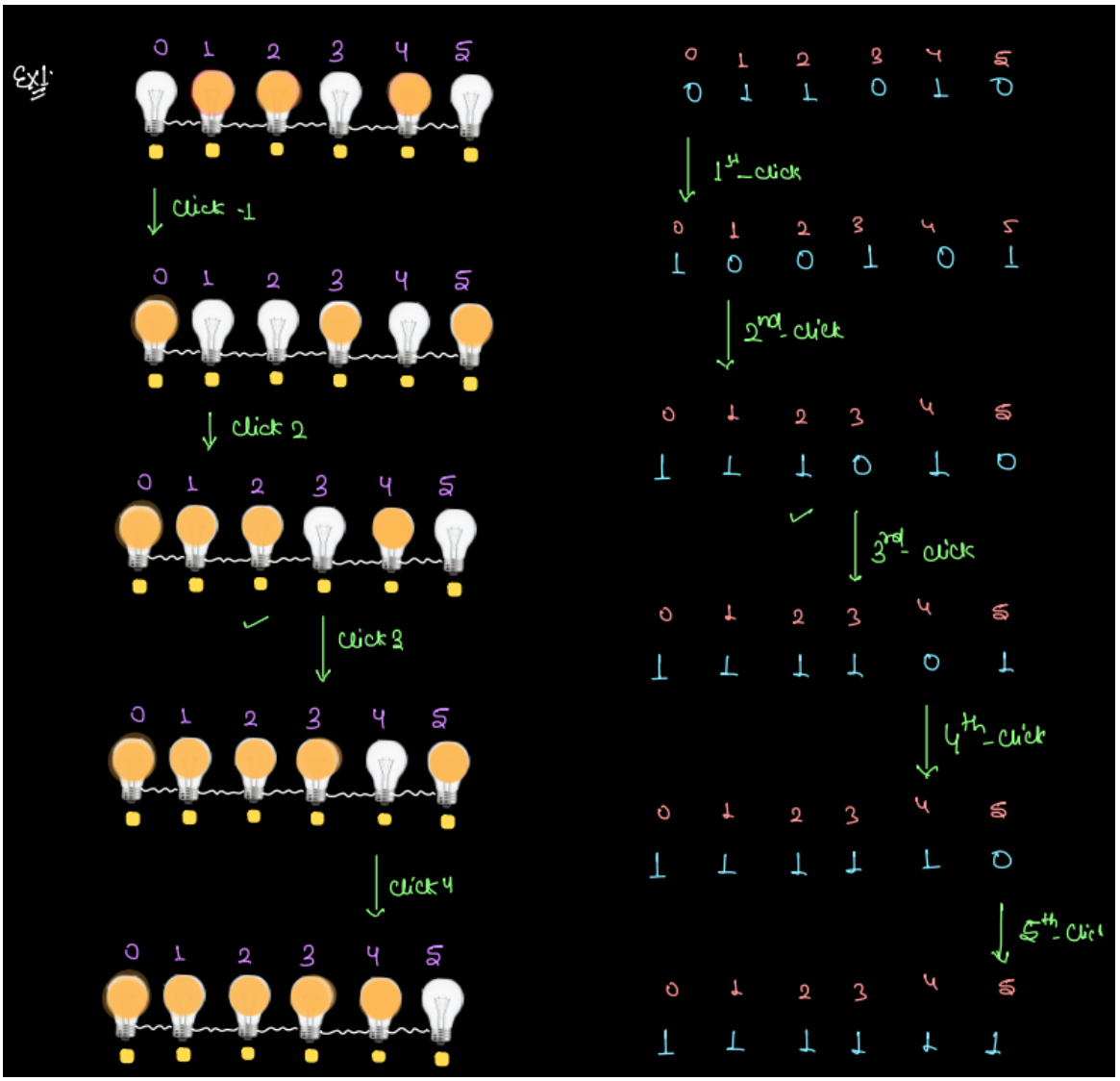

l1 l2 l3 l4 l5
arr → 3 2 4 5 2 7 1 15
0 1 2 3 4 5 6 7
leader count: 5
update in max
↳ leader cnt in array
max = 15
max = 3 → 4 → 5 → 7 → 15
index → 0 1 2 3 4 5
4 2 3 9 7 10
max=4 max=4 max=4 max=9 max=9 max=10
l=1 l=1 l=1 l=2 l=2 l=2
l=3

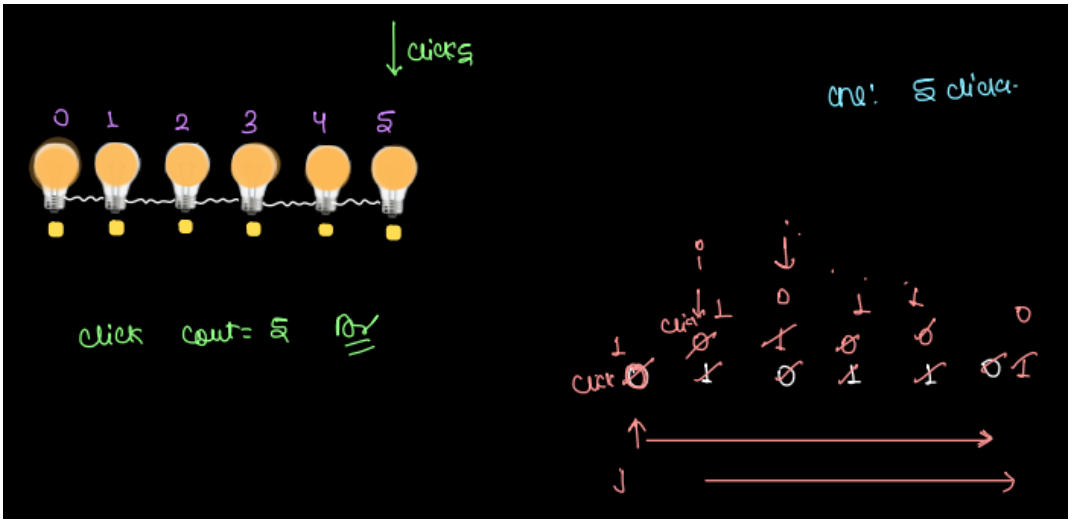

int leaderCount(int[] arr){
int n = arr.length;
int count = 1;
int max = arr[0];
for(int i=1; i<n; i++){
if(arr[i] > max){
max = arr[i];
count++;
}
}
return count;
}
T.C: O(n)
S.C: O(1)
Revision:
① b a g a g g
2 2 = ⑤
②
4 2 3 9 7 10
0 1 2 3 4 5
→ leader cnt: 3

- **N Bulbs**


Problem 3: N Bulbs
ON → arr[i]=1
OFF → arr[i]=0
Given N bulbs and their initial states, each bulb has a switch associated to it.
If we click on switch:
Every bulb on RIGHT including current bulb is flipped:
Problem ? Min. number of time we have to click on bulb so that every bulb will "ON".
Constraints : 1 <= N <= 10^5
N = 10^5 (max)
O(n) → max 9ir = 10^5 → manageable in 1 sec
O(n^2) : 10^5 x 10^5 = 10^10 → 1 sec X
O(n^2) → TLE


0 1 2 3 4 5
click
0 1 2 3 4 5
click
0 1 2 3 4 5
click
0 1 2 3 4 5


Ex1:
0 1 2 3 4 5
Click 1
Click 2
Click 3
Click 4
0 1 2 3 4 5
0 1 1 0 1 0
1st click
1 0 0 1 0 1
2nd click
1 1 1 0 1 0
3rd click
1 1 1 1 0 1
4th click
1 1 1 1 1 0
5th click
1 1 1 1 1 1




```
int minclicks(int[] arr){
    int n = arr.length;
    int c=0;
    for(int i=0; i<n; i++){
        if(arr[i]==0) {
            // bulb is off, need
               click to ON it
            c++;
            // flip all the bulb frm i to n-1
            arr[i] = 1;
            for(int j=i+1; j<n; j++){
                if(arr[j]==1) { arr[j]=0;}
                else { arr[j] = 1}
            }
        }
    }
    return c;
}
```

T.C: $O(n^2)$

S.C: $O(1)$

ON → 3time clicks

ON $\xrightarrow{C1}$ OFF $\xrightarrow{C2}$ ON $\xrightarrow{C3}$ OFF

ON → 4time clicks

ON $\xrightarrow{C1}$ OFF $\xrightarrow{C2}$ ON $\xrightarrow{C3}$ OFF $\xrightarrow{C4}$ ON

OFF → 3times clicks

OFF $\xrightarrow{C1}$ ON $\xrightarrow{C2}$ OFF $\xrightarrow{C3}$ ON

OFF → 4time

OFF $\xrightarrow{C1}$ ON $\xrightarrow{C2}$ OFF $\xrightarrow{C3}$ ON $\xrightarrow{C4}$ OFF

Observation:

① ON → Even clicks

ON (Initial State)

ON (final State)

ON $\xrightarrow{C1}$ OFF $\xrightarrow{C2}$ ON

② ON → odd clicks

ON (Initial State)

OFF (Final State)

	0	1	2	3	4	5
Initial State	OFF	OFF	ON	OFF	OFF	OFF
Click	F	F	F	F	F	F
	ON	ON	OFF	ON	ON	ON
			F	F	F	F
			ON	OFF	OFF	OFF
				F	F	F
				ON	ON	ON

OFF $\xrightarrow{\text{3 clicks}}$ ON

3 Flip

OFF $\xrightarrow{C1}$ ON $\xrightarrow{C2}$ OFF $\xrightarrow{C3}$ ON

ON $\xrightarrow{\text{2 Flip}}$ OFF

Initial State	Flip Counts	Final State
ON	odd	OFF → need a click
ON	even	ON
OFF	odd	ON
OFF	Even	off → need a click

	0	1	2	3	4	5
Initial State	OFF	OFF	ON	OFF	OFF	OFF
Click count 0	Even c=c+1 c=1	odd c=1	odd c=c+1 c=2	Even c=c+1 c=3	odd c=3	odd c=3
Final State	ON	ON	ON	ON	ON	ON

i

Click count : 1

n=5

0 1 1 0 1
0 1 2 3 4

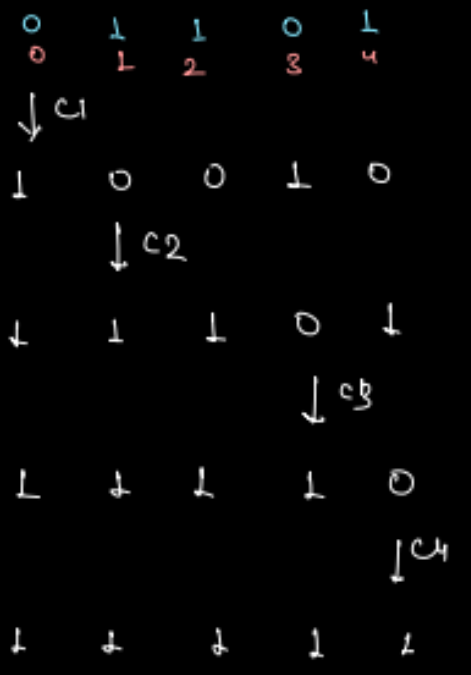
```
int clickCount(int[] arr) {
    int n = arr.length;
    int c = 0;
    for(int i = 0; i < n; i++) {
        if( arr[i] == 1 && c%2 == 1) { c = c+1; }
        else if( arr[i] == 0 && c%2 == 0) { c = c+1; }
    }
    return c;
}
```

Initial ON — Click count Odd

Initial OFF — Click count Even

c = 0 1 2 2 4 → all the bulbs on

T.C: O(n)
S.C: O(1)



double: neglect lower order term.

$f(n)$ $n^2 + \underbrace{10n}$

% contribution of $10n$ in n^2+10n

$$= \frac{10n}{n^2+10n} \times 100$$

7 Apples + 3 oranges ⇒ % of orange in total fruits.

$$= \frac{3}{10} \times 100 = 30\%$$

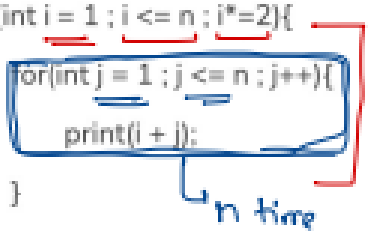
Sum of G.P $= \frac{a(r^n-1)}{r-1}$ $\quad a=2,\ r=2$

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

Find Time Complexity - 8

What is the time complexity of the following code snippet

```
for(int i = 1 ; i <= n ; i*=2){
    for(int j = 1 ; j <= n ; j++){
        print(i + j);
    }
}
```



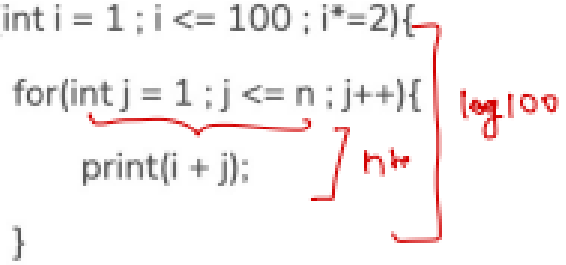
time → n logn

O(n logn)

Find Time Complexity - 6

What is the time complexity of the following code snippet

```
for(int i = 1 ; i <= 100 ; i*=2){
    for(int j = 1 ; j <= n ; j++){
        print(i + j);
    }
}
```



log100 × n

constant

= O(n)

C++ | Java | Python

```
for (i = 0; i < N; i++) {
  for (j = i; j < N; j++) {
    break;
  }
}
```

N

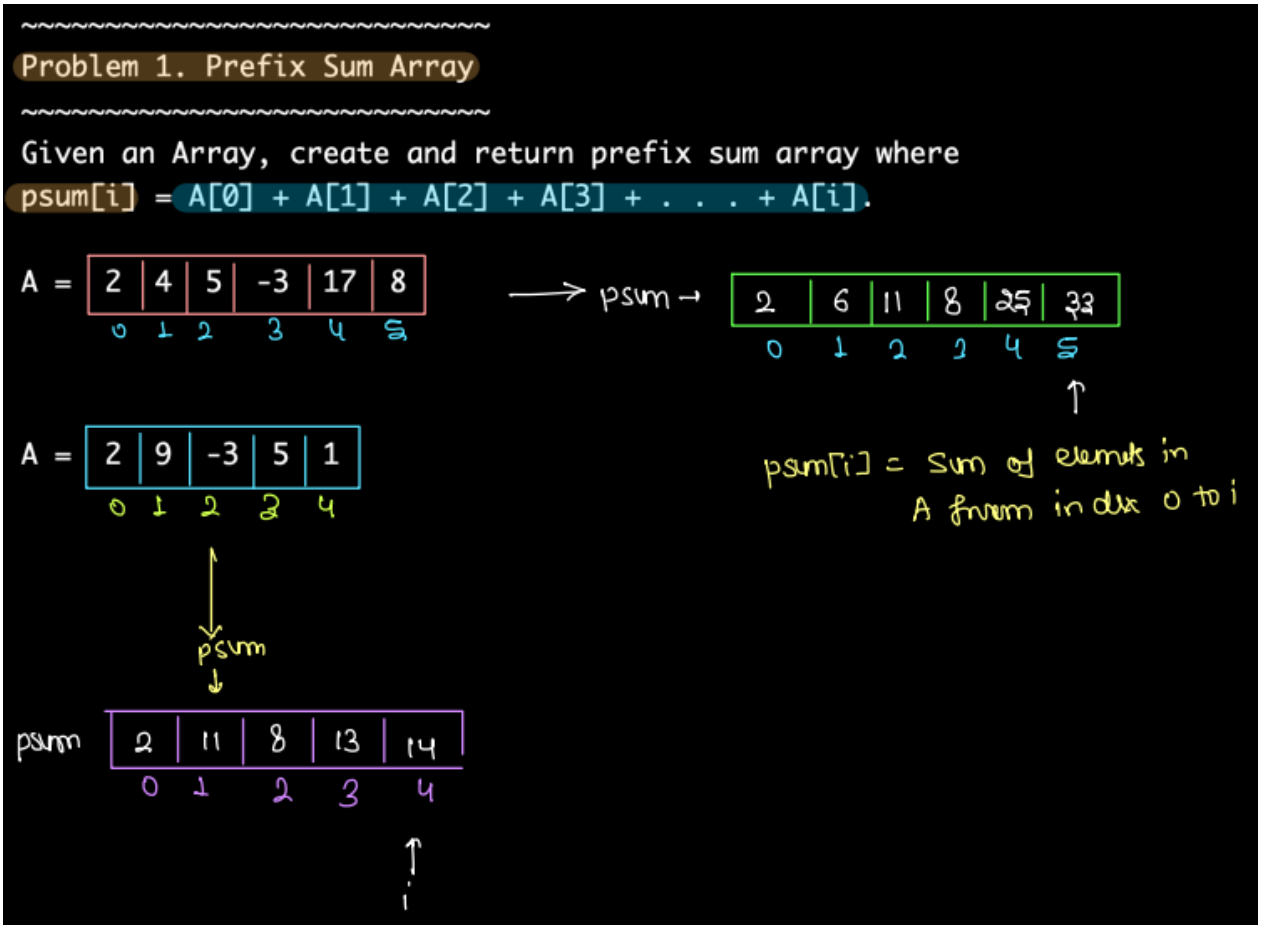
O(1)

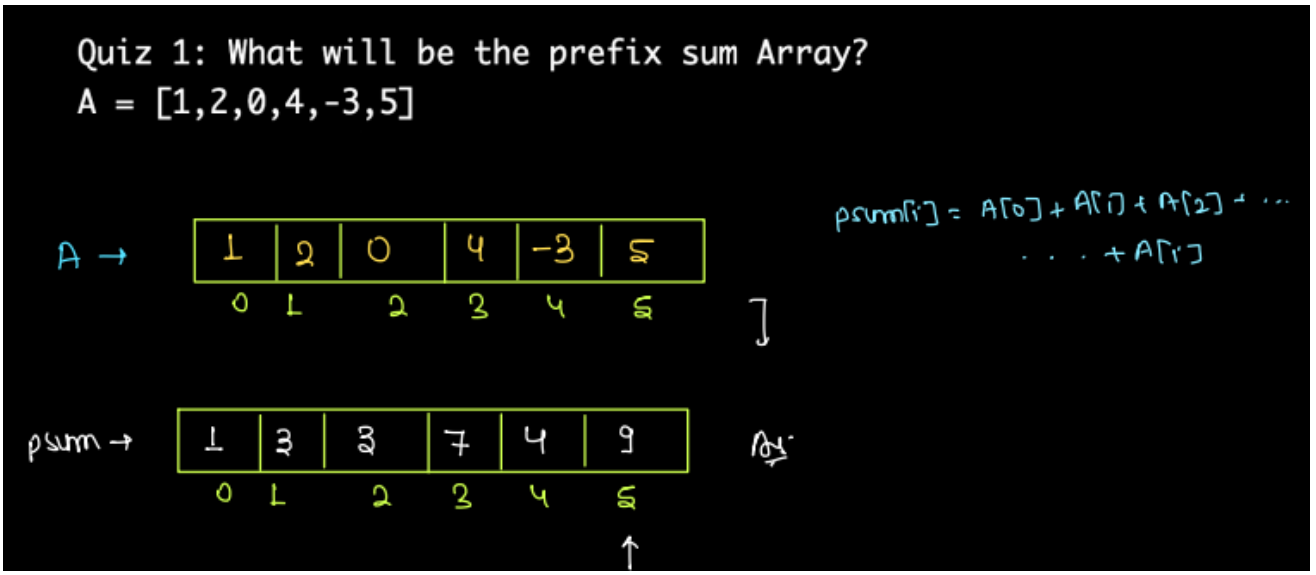
O(n)

DSA: Prefix Sum - 27 - Apr 2023

- Prefix Sum Array
- Range Sum Query
- Equilibrium Index
- Even Number for Q query

- **Prefix Sum Array**





observations:

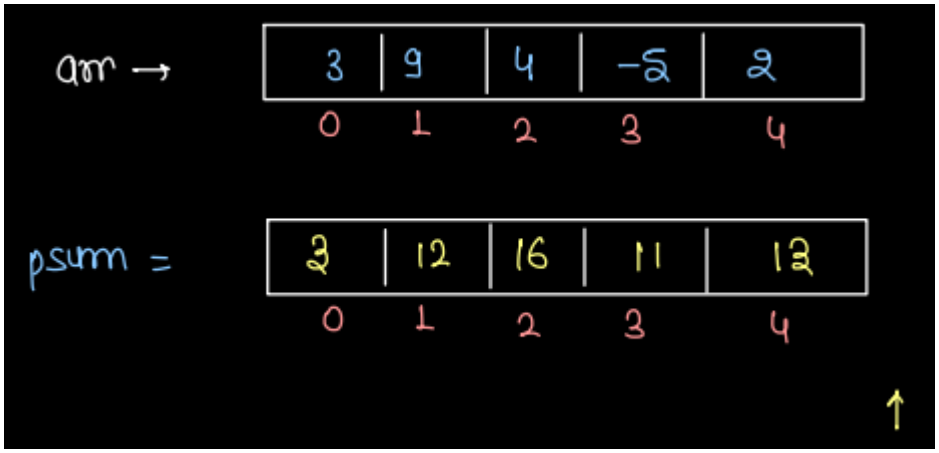
psum[0] = A[0]

psum[1] = A[0] + A[1] ⇒ psum[0] + A[1]

psum[2] = A[0] + A[1] + A[2] ⇒ psum[1] + A[2]

psum[3] = A[0] + A[1] + A[2] + A[3] ⇒ psum[2] + A[3]

$$psum[i] = psum[i-1] + A[i]$$




```
int[] prefixSum(int[] arr){
    int n = arr.length;
    int[] psum = new int[n];
    // fill prefix sum array.
    // fill psum[0] first
    psum[0] = arr[0];
    for(int i=1; i<n; i++){
        psum[i] = psum[i-1] + arr[i]
    }
    return psum;
}
```

n = 4

arr: 3 | 9 | 4 | -5 (0 1 2 3)

psum: 3 | 12 | 16 | 11 (0 1 2 3)

i: loop stop

Time Complexity: O(n)

Space Complexity: Since problem is asking you to return one array of size 'n'
→ that space not consider in S.C.

S.C: O(1)

```
int[] prefixSum(int[] arr){
    int n = arr.length;
    int[] psum = new int[n];
    // fill prefix sum array.
    // fill psum[0] first
    psum[0] = arr[0]
    for(int i=1; i<n; i++){
        psum[i] = psum[i-1] + arr[i];
    }
    return psum;
}
```

→ Requirements of problem.

not consider in space complexity

n: 4 byte

i : 4 byte

total space : 8 byte

S.C = O(1)

- **Range Sum Query**

Problem 2. Range Sun queries

Given an array A[] of size N and Q queries. Find the answer for all queries in the given range.

Constraints :

1 <= n <= 10^5

1 <= Q <= 10^5

In single query, L-R

find sum from L-index to R-index.

A =

0	1	2	3	4	5	6	7
3	4	-2	6	8	10	13	1

Q = 4 (L <= R)

query:

0th (Left)	1st (Right)
1	3
2	6
5	5
0	3

	L	R	ans
I	1] ⟷	3] ⟶	8
II	2] ⟷	6] ⟶	85
III	5] ⟷	5] ⟶	10
IV	0] ⟷	3] ⟶	11

left for ith query = query[i][0]

right for ith query = query[i][1]

Quiz 2: Given the scores of the 10 overs of a cricket match

2, 8, 14, 29, 31, 49, 65, 79, 88, 97

1 2 3 4 5 6 7 8 9 10

How many runs were scored in just 7th over?

0	1	2	3	4	5	6	7	8	9
2	8	14	29	31	49	65	79	88	97
1	2	3	4	5	6	7	8	9	10

Score in end of 7th over = 65

Score in end of 6th over = 49

Runs scored in 7th over = 65 - 49 = 16 Ans

Quiz 3: Given the scores of the 10 overs of a cricket match
[2, 8, 14, 29, 31, 49, 65, 79, 88, 97]
How many runs were scored from [6th over - 10th over]

runs scored in ending of 10th over = 97

runs scored in ending of 5th over = 31

difference = 97 - 31

= 66 Ans

Problem 2. Range Sun queries

Given an array A[] of size N and Q queries. Find the answer for all queries in the given range.

Constraints :

1 <= n <= 10^5 → 10^5

1 <= Q <= 10^5 → 10^5

T.C: O(n*q) ⇒ $10^5 * 10^5$ ghr

⇒ 10^{10} ghr

⇒ 1 sec = 10^8 ghr

takes more than one sec.

TLE.

O(n+q) = $10^5 + 10^5 = 2*10^5$ ⇒ ✓ 1sec → O(n+q) perfect Approach

A =

0	1	2	3	4	5	6	7
3	4	-2	6	8	10	13	1

Q = 4 (L <= R)

L	R	ans
1	3	8
2	6	35
5	5	10
0	3	11

Brute force Approach:

① Every possible query, Extract L & R from query.

② find sum from L-index to R-index for that query.

```
int[] solve(int[] A, int[][] query) {
    int n = A.length;
    int q = query.length;
    int[] ans = new int[q];
    for(int i = 0; i < q; i++){
        int l = query[i][0];
        int r = query[i][1];
        // calculate sum b/w l to r.
        & fill it in ans[i]
        int sum = 0;
        for(int j = l; j <= r; j++){
            sum += A[j];
        }
        ans[i] = sum;
    }
    return ans;
}
```

In worst case: Possible value of l & r is 0 to n-1

q times x n time

total: q x n

T.C: O(q * n)

Improvement!

psum[6] = sum(0 to 6) in A

$$sum(3,6) = sum(0,6) - sum(0,2)$$

$$= psum[6] - psum[2]$$

$$psum[6] = A[0] + A[1] + A[2] + A[3] + A[4] + A[5] + A[6]$$

$$psum[2] = A[0] + A[1] + A[2]$$

$$= A[3] + A[4] + A[5] + A[6]$$

sum from 3 to 6

TODO:

$Sum(2,5) = Sum(0,5) - Sum(0,1)$

$psum[5] - psum[1]$

$psum[5] = A[0] + A[1] + A[2] + A[3] + A[4] + A[5]$

$psum[1] = A[0] + A[1]$

$= A[2] + A[3] + A[4] + A[5]$

left

$Sum(0,3) = psum[3] - psum[-1]$ ✗

↳ psum[3]

$Sum(L,R) = psum[R] - psum[L-1]$ where $L \neq 0$

```
Sum(L,R):
    if(L==0){
        ans → psum[R];      O(1)
    }else{
        ans → psum[R] - psum[L-1]
                            O(1)
    }
```

if psum is already calculated.

T.C.: O(1) for single query

```
int[] solve(int[] A, int[][] query){
    int n = A.length;
    int q = query.length;
    // generate prefix sum.
    int[] psum = prefixSum(A);          → T.C.1: O(n)
    // solve every query
    int[] ans = new int[q];
    for(int i=0; i<q; i++){             → T.C.2: O(q)
        int l = query[i][0];
        int r = query[i][1];
        if(l==0){
            ans[i] = psum[r];
        }else{
            ans[i] = psum[r] - psum[l-1];
        }
    }
    return ans;
}
```

this space is part of space complexity because we are making it for using.

Total T.C.: O(n+q)

Space comp. SC: O(n)

Break time: 8:30 - 8:40

- **Equilibrium Index**

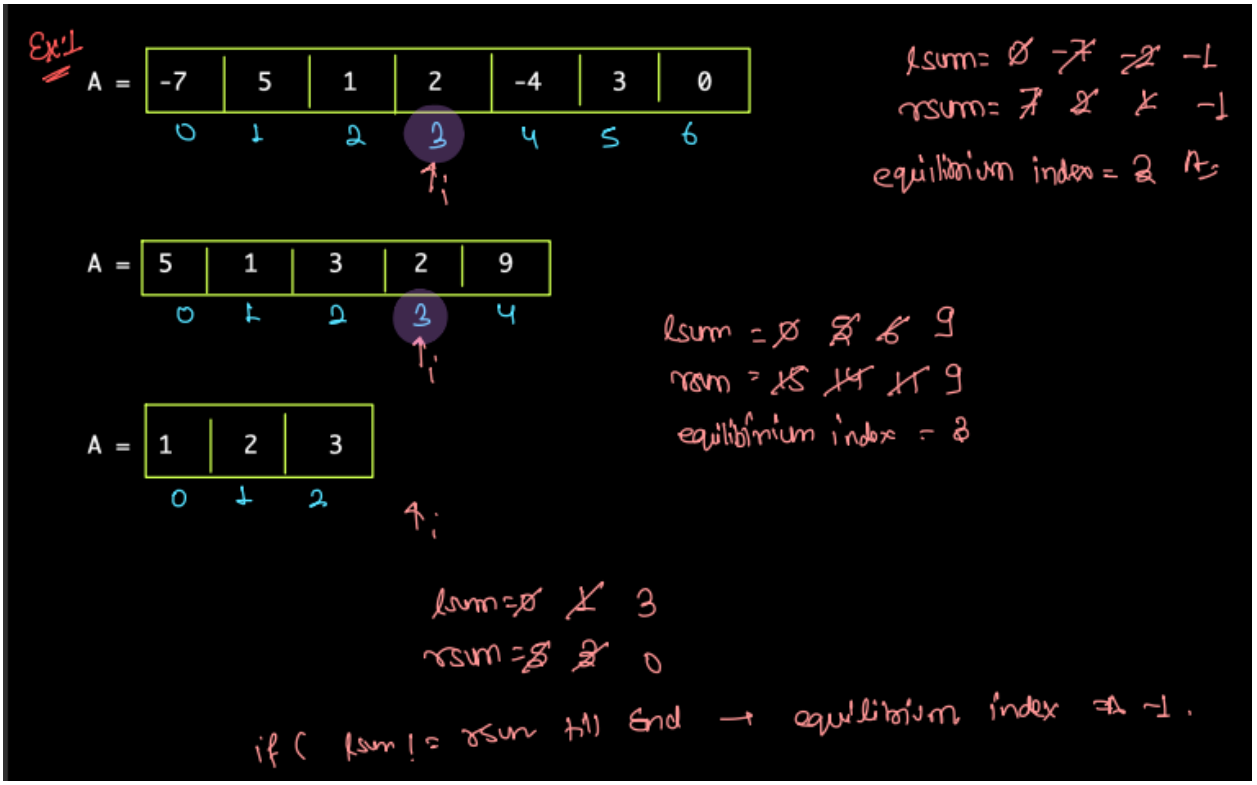
tag : Amazon, Adobe

Problem 3 : Equilibrium Index

Given an array A[] of size N, Find the equilibrium index.

i is an equilibrium index when :

Sum of all the elements in left side of i i = Sum of all the elements in right side of i } i is equilibrium index



brute force Approach :

```
for(int i=0; i<n; i++) {
    // calculate left sum
    for(int j=0; j<i; j++){ lsum += A[j] }
    // calculate right sum.
    for(int j=i+1; j<n; j++) { rsum += A[j]}
    if( lsum == rsum)
        return i;
}
return -1;
```

Steps:

1. create an every index of array.
2. for every possible array calculate left sum & right sum
3. if lsum & rsum is equal return index.
4. return -1;

TODO:

H/W

i	Itr by ①	Itr by ②	Total Itr.

```
for(int i=0; i<n; i++) {
    // calculate left sum
    for(int j=0; j<i; j++){ lsum += A[j] }
    // calculate right sum.
    for(int j=i+1; j<n; j++) { rsum += A[j] }
    if( lsum == rsum )
        return i;
}
return -1;
```

T.C: $O(n^2)$

Optimisation:

Sum of all the Elements which are left of i = Sum of all Elements on right side of i

psum[i-1] = psum[n-1] - psum[i]

```
int equilibriumIndex(int[] A) {
    int[] psum = prefixSum(A);   // n iters, n space
    int n = A.length;
    for(int i=0; i<n; i++) {     // n iters.
        int lsum = 0;
        if( i > 0){
            lsum = psum[i-1];
        }
        int rsum = psum[n-1] - psum[i];
        if( lsum == rsum) {
            return i;
        }
    }
    return -1;
}
```

T.C: O(n)
S.C: O(n)

5	1	3	2	9
0	1	2	3	4

psum =

5	6	9	11	20
0	1	2	3	4

↑

i	lsum	rsum
0	0	15
1	5	14
2	6	11
3	9	9

o/p: 3 it is eq. index.

Quiz 4: Sum of all elements from [i+1 ,N-1] using pf[]

L R

Sum(2,4) => psum[4] - psum[1]

psum[R] - psum[L-1]

psum[N-1] - psum[i+1-1] => psum[N-1] - psum[i]

Quiz 5: ar[7] = {-7 1 5 2 -4 3 0 }
How many equlibrium indices are there ?

	-7	1	5	2	-4	3	0
Index	0	1	2	3	4	5	6
Psum	-7	-6	-1	1	-3	0	0
lsum	0	-7	-6	-1	1	-3	0
rsum	7	6	1	-1	3	0	0

0-(-7) = 7

count : ~~0~~ 2

2, 6 → are equilibrium index

count : 2

lsum = 0
if(i > 0)
lsum = psum[i-1]
}
rsum = psum[n-1] - psum[i]

psum[n-1] - psum[i]
= 0 - 0 = 0

0 - (-7) = 7

Basic pattern:

a - (-b) = a + b

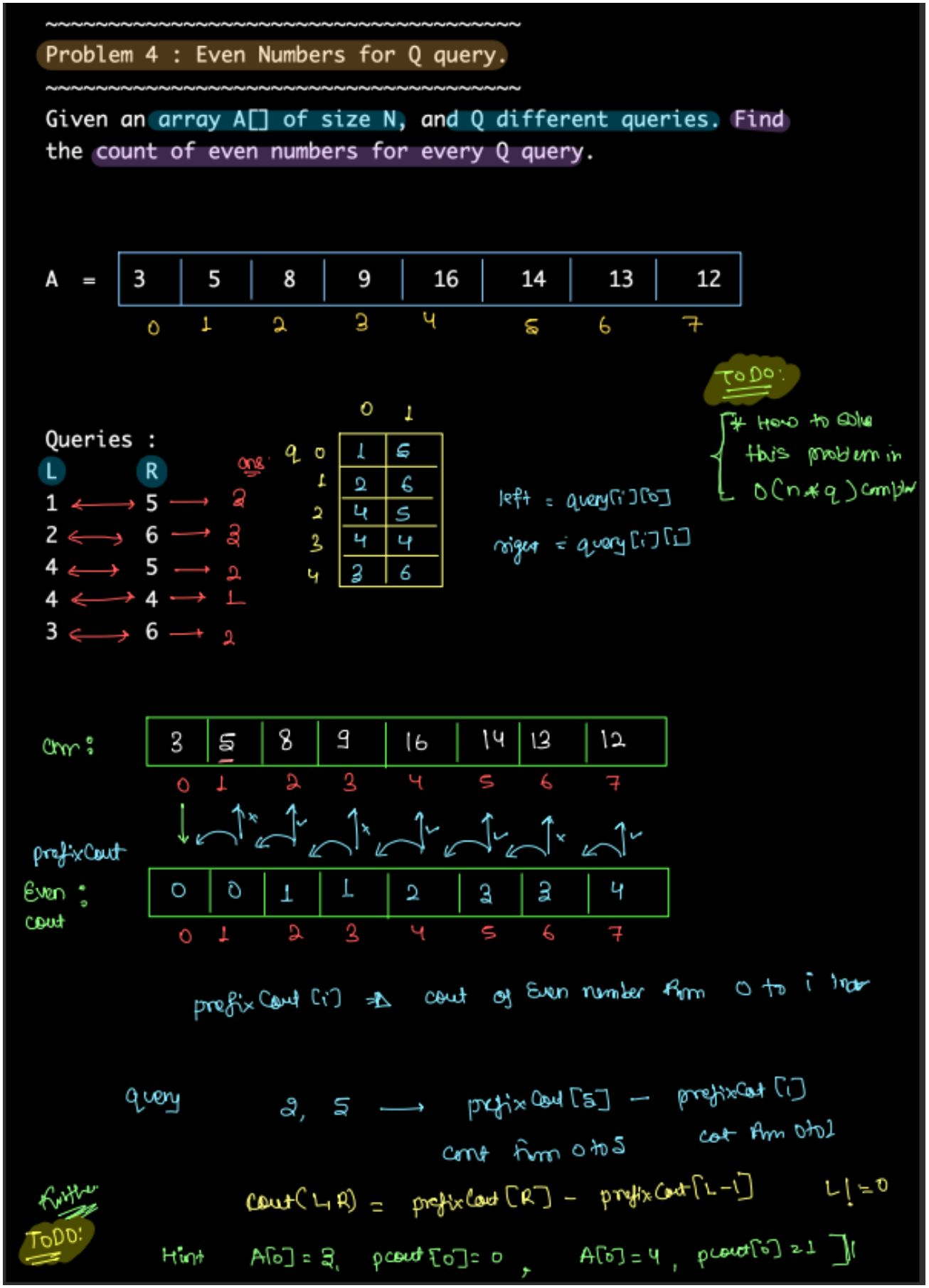
a - b = a - b

a - (+b) = a - b

-a - b = -(a+b)

-2 - 3 = -(2+3)
= -5

- **Even Number for Q query**



1. Prefix Sum -> means we have to get from start and sum it and store in ith position like

prefixSum[i] = prefixSum[i-1] + A[i]

int[] A = {1, 2, 3, 4, 5};

int[] prefixSum = new int[A.length]

prefixSum[0] = A[0]

first value of array is [0] position and array will start from 1 index

prefixSum[1] = A[0] + A[1];

prefixSum[2] = A[0] + A[1] + A[2];

prefixSum[3] = A[0] + A[1] + A[2] + A[3];

prefixSum[4] = A[0] + A[1] + A[2] + A[3] + A[4];

prefixSum[5] = A[0] + A[1] + A[2] + A[3] + A[4] + A[5];

2. Q queries -> Find the answer for Q queries

Q queries menas 2D array of 2 x 2 where

- first column is L and second Column is R
- L ith query is query[i][0];
- R ith query is query[i][1];

if we have to get sum of (2, 5) then we will do sum[5]-sum[1] like score of over mathematical calculator

sum of 5 -> A[0] + A[1] + A[2] + A[3] + A[4] + A[5]

sum of 1 -> A[0] + A[1]

now sum of 2,5 is left

Note -> If we have to calculate sum(0, 3) in this case we will take sum of right index instead doing sum[3] - sum[-1]

here is the genrlize formula for

sum(L,R) = sum[R] - sum[L-1];

Left can not be 0 elase only sum(L,R) = sum[R] no left;

3. Equilibrium index of an array -> Equilibrium index is when ith left values and ith right values sum of equiel it called equilibrium index.

Brute force approach

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

// get L sum using for loop i = 0 to i < i

// get R sum using for loop i+1 till i<n

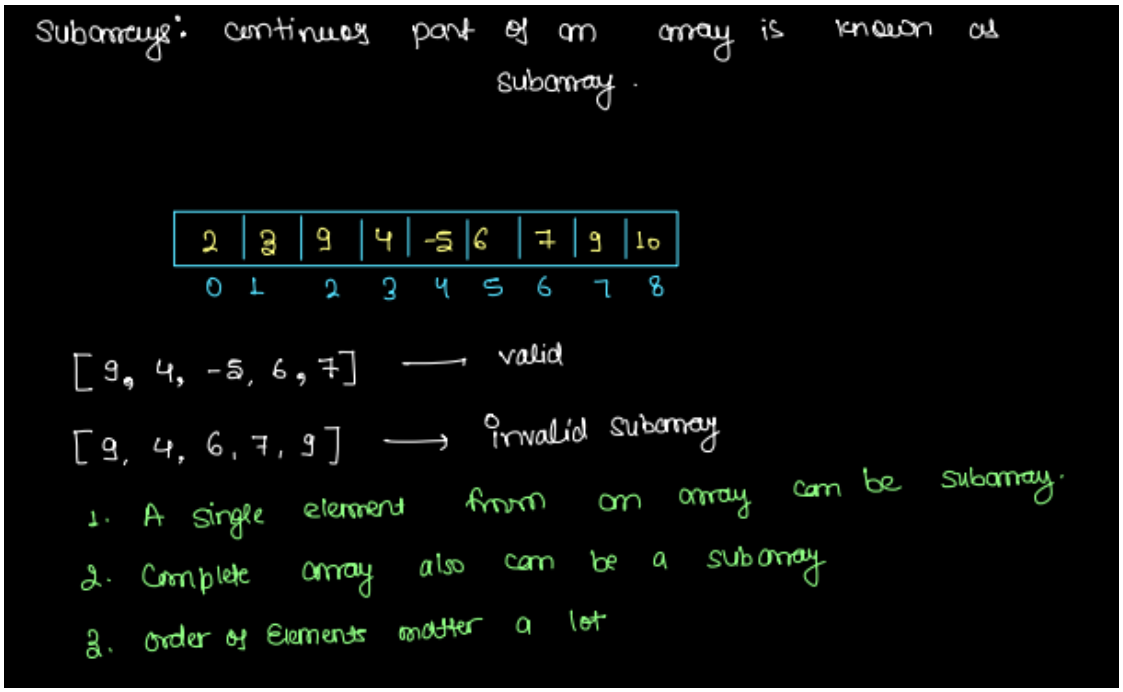
compair both sum and return the result

}

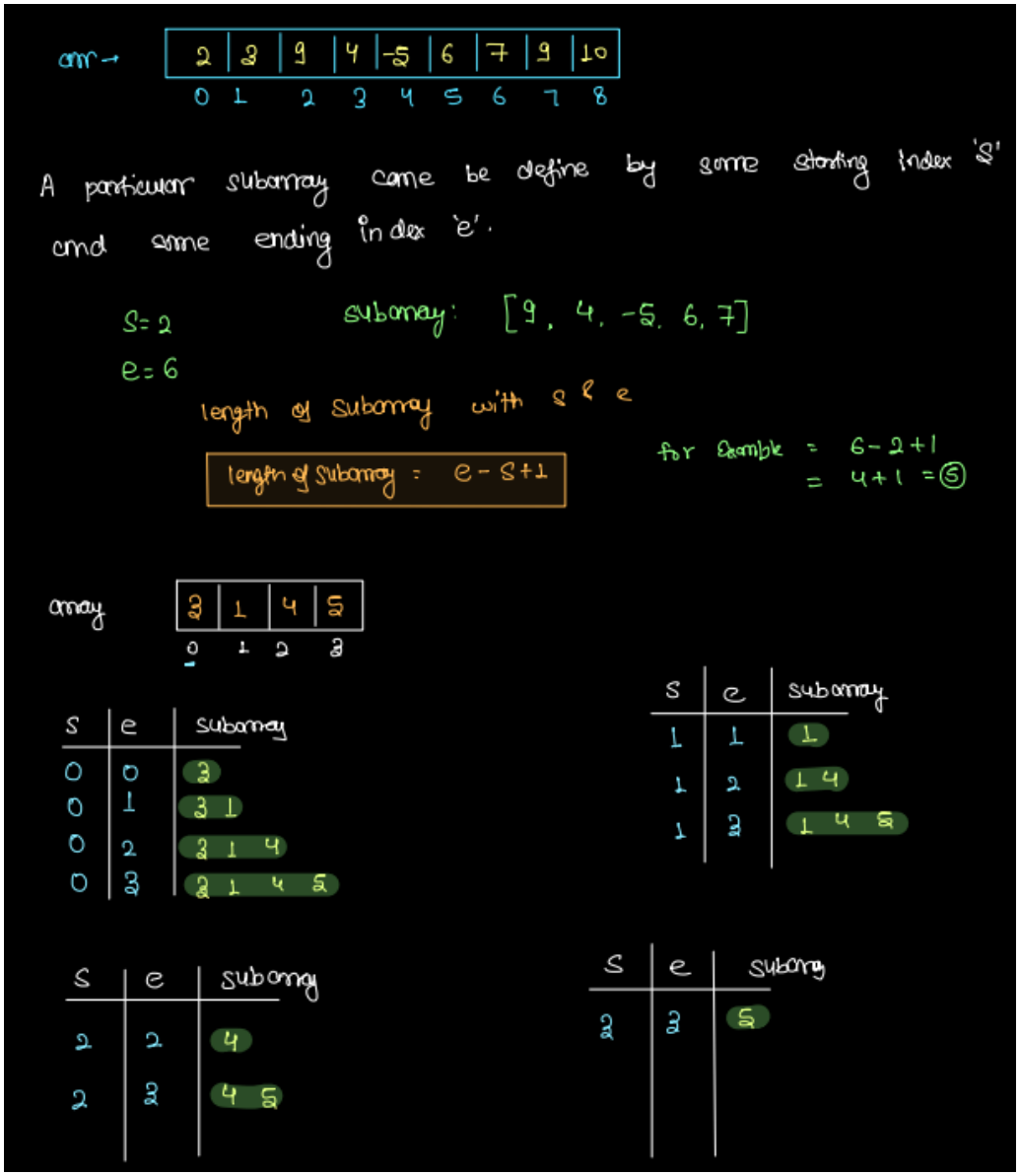
DSA: Sub Array - 29 - Apr 2023

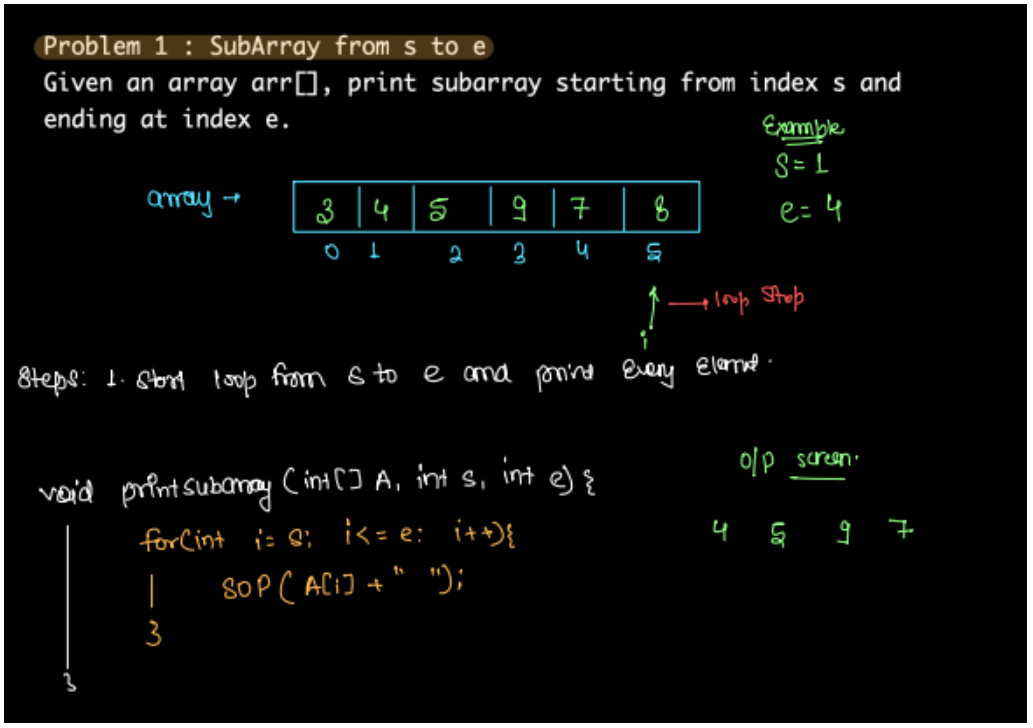
- **Introduction of subarray**
- **Subarrar from s to e**
- **Print All subarray**
- **Sum of every subarray**
- **sum of all subarray's sum**

Introduction of subarray

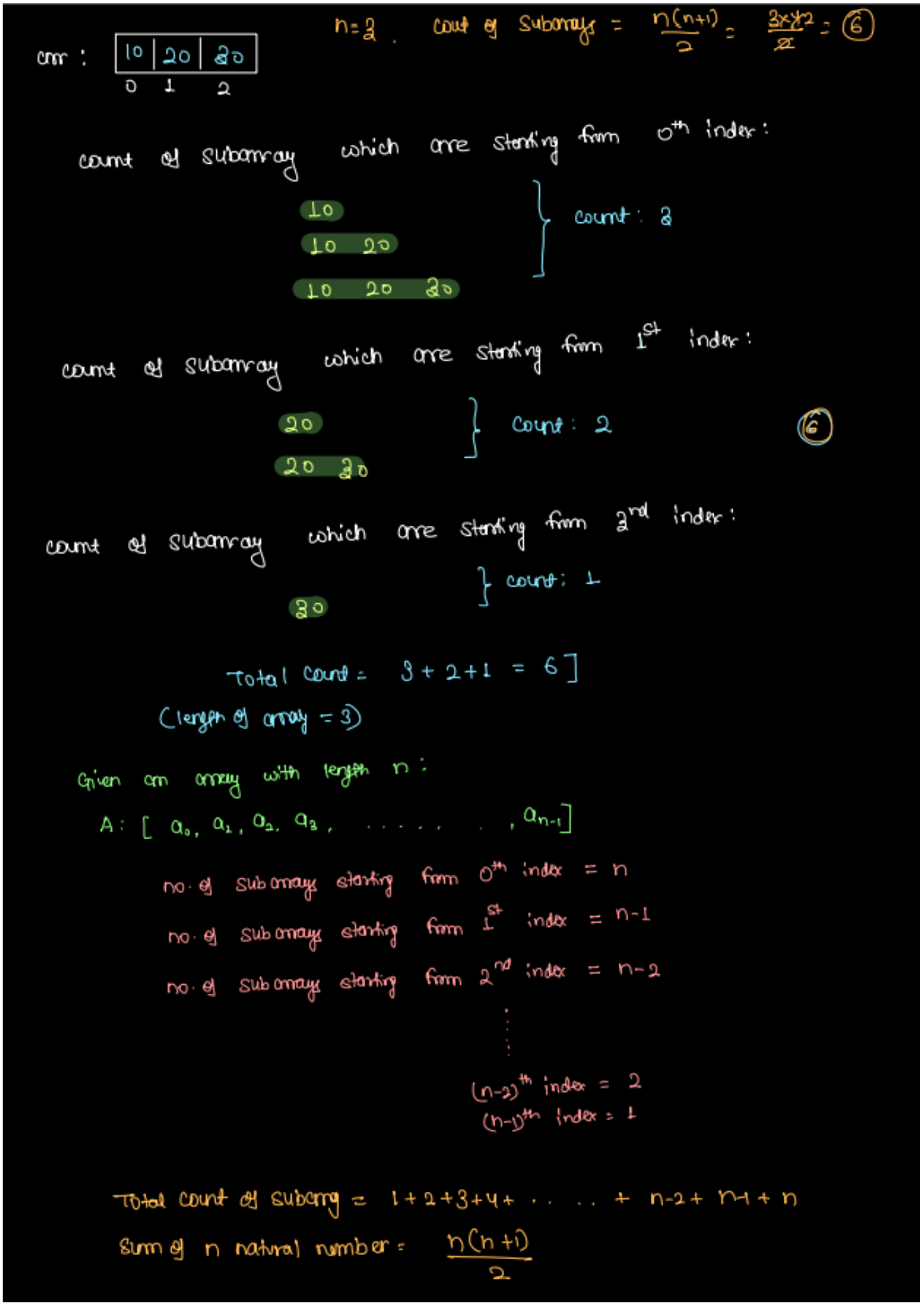


Subarrar from s to e





Print All subarray



Problem 2 : Print All subarrays

Given an array arr[], print all the subarrays possible in array arr

3	-1	2
0	1	2

total subarray = $\frac{n(n+1)}{2}$

$= \frac{3 \times 4}{2} = 6$

S = 0

(0,0) - 3

(0,1) - 3 -1

(0,2) - 3 -1 2

S = 1

(1,1) - -1

(1,2) - -1 2

S = 2

(2,2) - 2

i = 0 to n-1

j = i to n-1

S = 0 → e = 0, 1, 2

S = 1 → e = 1, 2

S = 2 → e = 2

```
void printAllSubarrays(int[] A){
    int n = A.length;
    for(int s = 0; s<n; s++){
        for(int e = s; e<n; e++){
            // we have all possibility of
               's' and 'e' is here
            for(int i=s; i<=e; i++){
                System.out.print(A[i] + " ");
            }
            System.out.println();
        }
    }
}
```

T.C: $O(n^3)$

S.C: $O(1)$

n = 3

S = 0 → e = 0, 1, 2

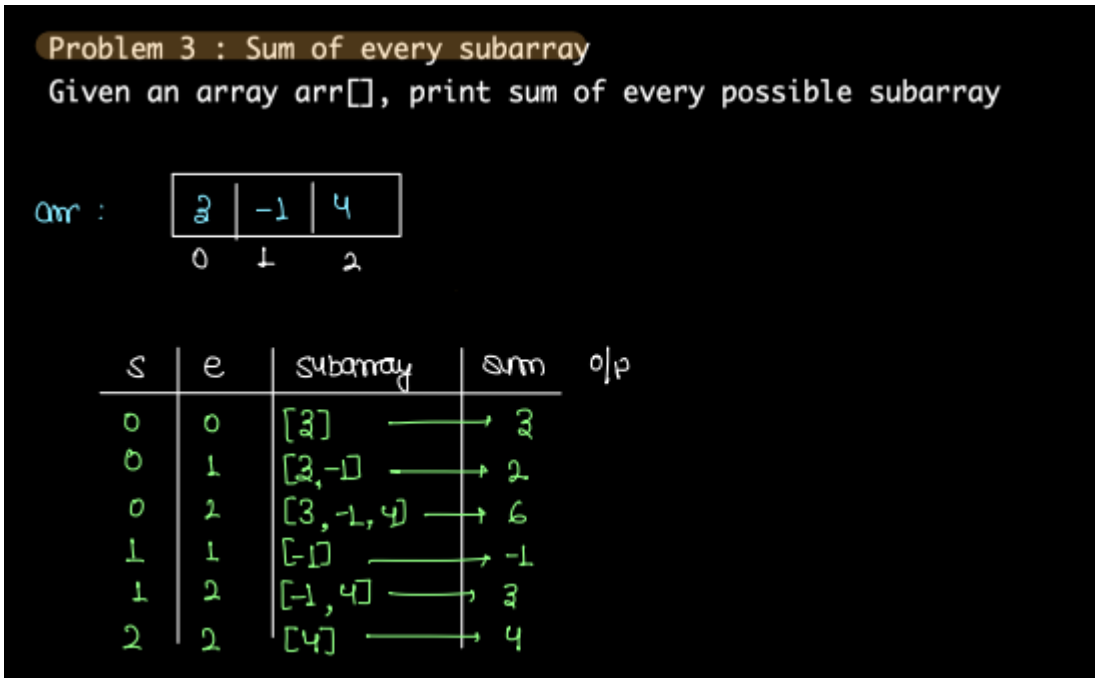
S = 1 → e = 1, 2

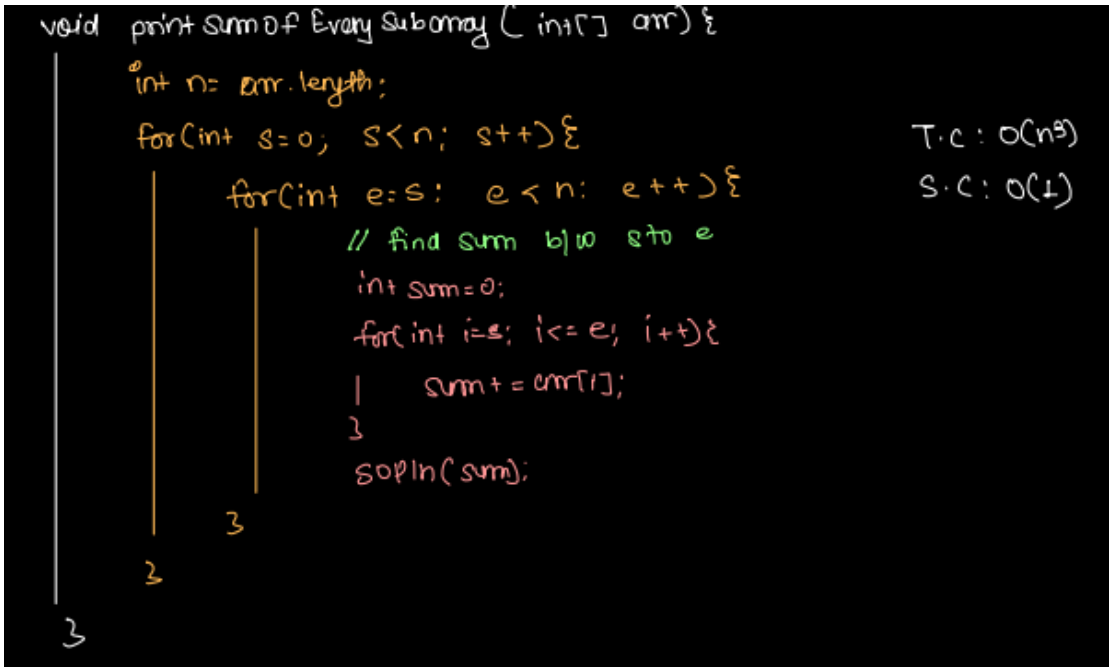
S = 2 → e = 2

arr: 3 (0) -1 (1) 2 (2), n = 3

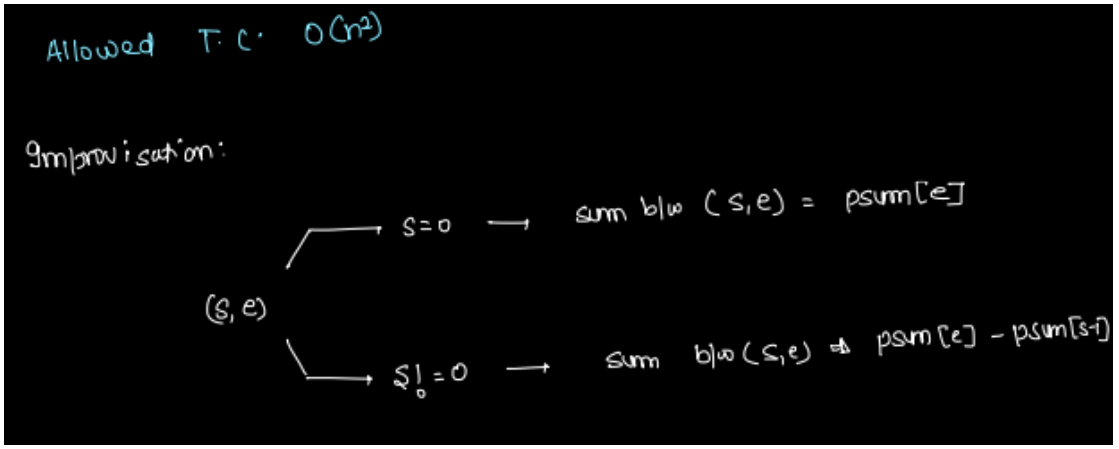
s	e	subarray
0	0	(0,0) → 3
	1	(0,1) → 3 -1
	2	(0,2) → 3 -1 2
1	1	(1,1) → -1
	2	(1,2) → -1 2
2	2	(2,2) → 2

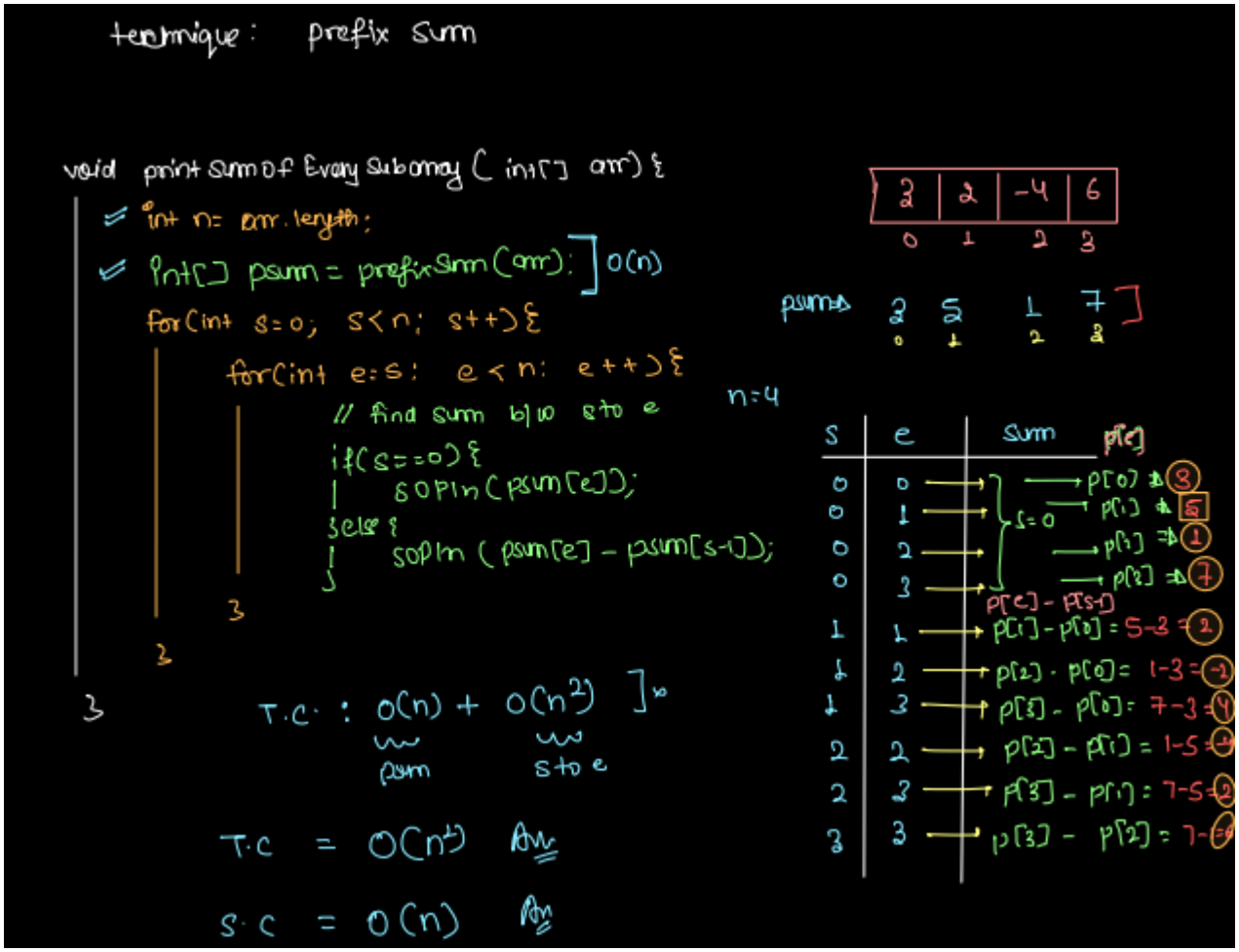
Sum of every subarray



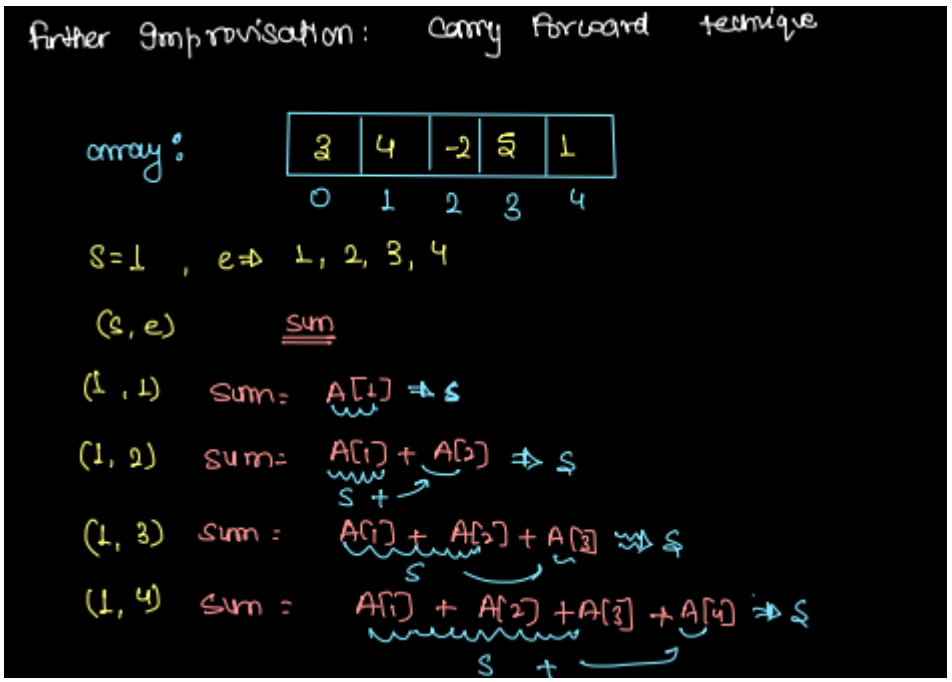


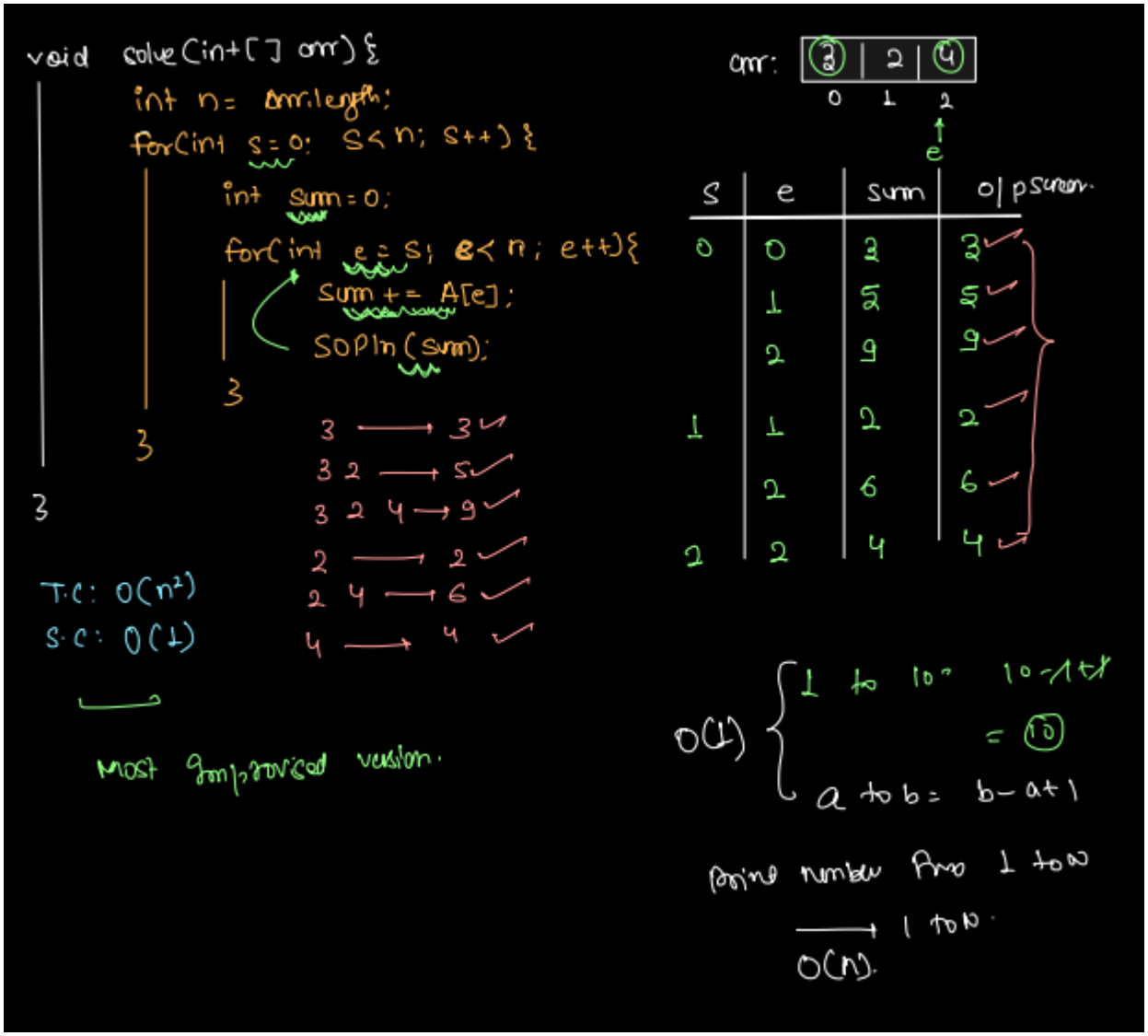
Improvisation





Can we reduce space complexity??





Contribution technic

Proble4: Count of all possible Sub Array ->

count of All possible subarrays which have A[i] available

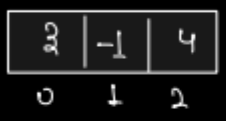
any valid starting point can be any ith row but ending point can not be valid before i like 0 1 2 3 4 5

for 3 ending point can start from 4

sum = x*A[0] + y*A[1] + Z*A[2]

Problem 4 : Sum of all subarrays sum.

Given an array arr[], print sum of all subarrays's sum.



S	e	Subarray	sum
0	0	[3]	3
0	1	[3, -1]	2
0	2	[3, -1, 4]	6
1	1	[-1]	-1
1	2	[-1, 4]	3
2	2	[4]	4

o/p:

s/p: 3 + 2 + 6 + (-1) + 3 + 4

= 17 Ans

brute force approach:

```
int solve(int[] arr) {
    int ans = 0;
    int n = arr.length;
    for(int s = 0; s < n; s++) {
        int sum = 0;
        for(int e = s; e < n; e++){
            sum += A[e];
            // SOPln(sum);
            ans += sum;
        }
    }
    return ans;
}
```

T.C. : $O(n^2)$

S.C : $O(1)$

Expected T.C. : $O(n)$

S.C. : $O(1)$

s	e	Subarray	Sum
0	0	[3]	3
0	1	[3, -1]	2
0	2	[3, -1, 4]	6
1	1	[-1]	-1
1	2	[-1, 4]	3
2	2	[4]	4

A[0]

A[0] + A[1]

A[0] + A[1] + A[2]

A[1]

A[1] + A[2]

A[2]

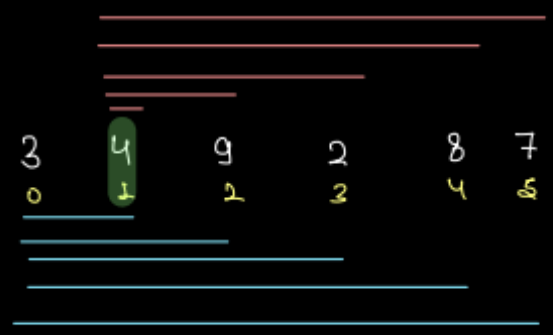
$3 * A[0] + 4 * A[1] + 3 * A[2]$

$3 * 3 + 4 * (-1) + 3 * 4$

$= 9 - 4 + 12 = 17$ Ans

Sum: $x * A[0] + y * A[1] + z * A[2]$

How many time i^{th} element is coming in every Sum.



10 different subary.

A[4] is present

valid Ending pair => e : i, i+1, i+2 . . . . , n-1

Group Boys Group Girls

Different possibility for pair up:

Starting pt Ending

i+1 n-i

count of ith index in overall sum of subarray = (i+1) * (n-i)

```
int sum(int[] arr){
    int n= arr.length;
    int ans=0;
    for(int i=0; i<n; i++){
        ans += arr[i] * [(i+1) * (n-i)]
    }
    return ans;
}
```

T.C. O(n)

S.C. O(1)

2	-1	4
0	1	2

n=3

i	arr[i] * (i+1)(n-i)
0	3 * (1) * (3) = 9
1	(-1) * (2) * (2) = -4
2	4 * (3) * (1) = 4

ans = 17

contribution technique :


```
int sum(int[] arr){
    int n= arr.length;
    int ans=0;
    for(int i=0; i<n; i++){
        ans += arr[i] * [(i+1) * (n-i)]
    }
    return ans;
}
```

2	-1	4
0	1	2

n=3

i	arr[i] * (i+1)(n-i)
0	3 * (1) * (3) = 9
1	(-1) * (2) * (2) = -4
2	4 * (3) * (1) = 4

ans = 17

Notes -> DSA: Prefix & Subarrays - 4 - May 2023

Imp Observation of prefix sum —----------

Continuous Sum Query —----------

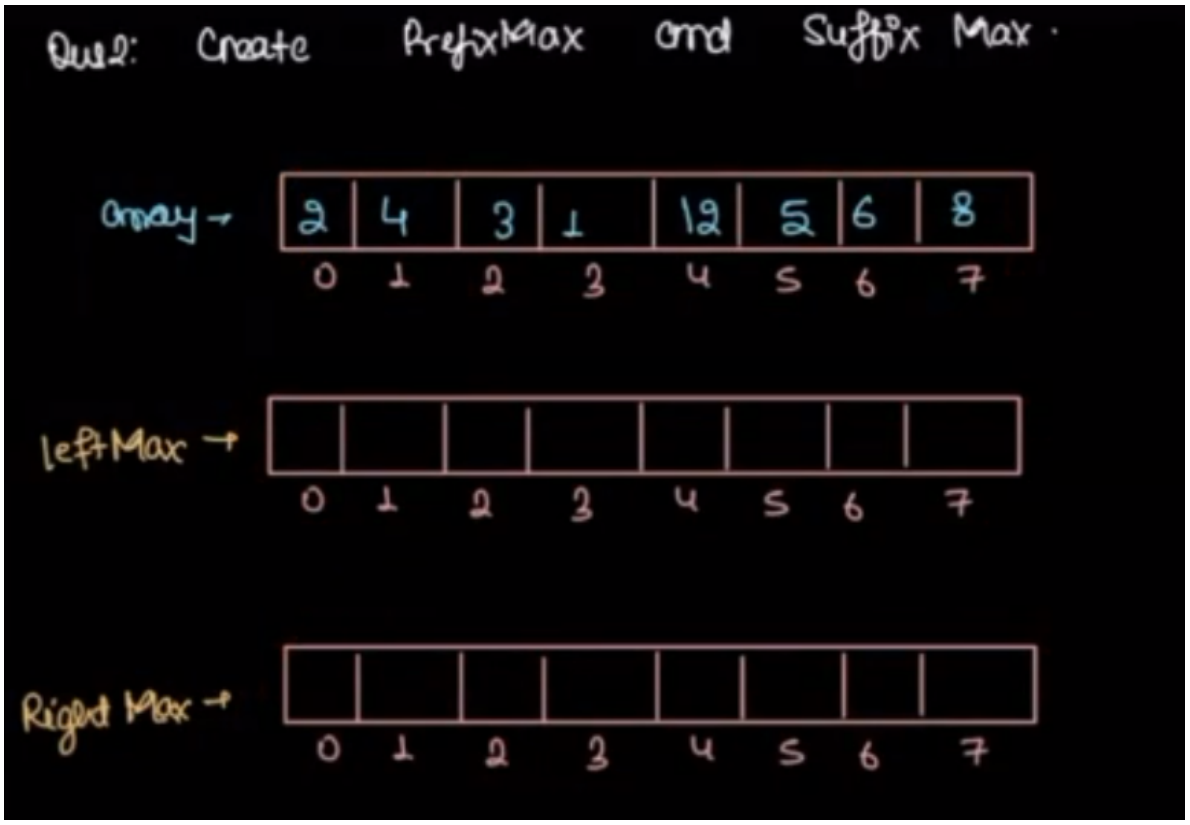
Leftmax and Rightmax —----------

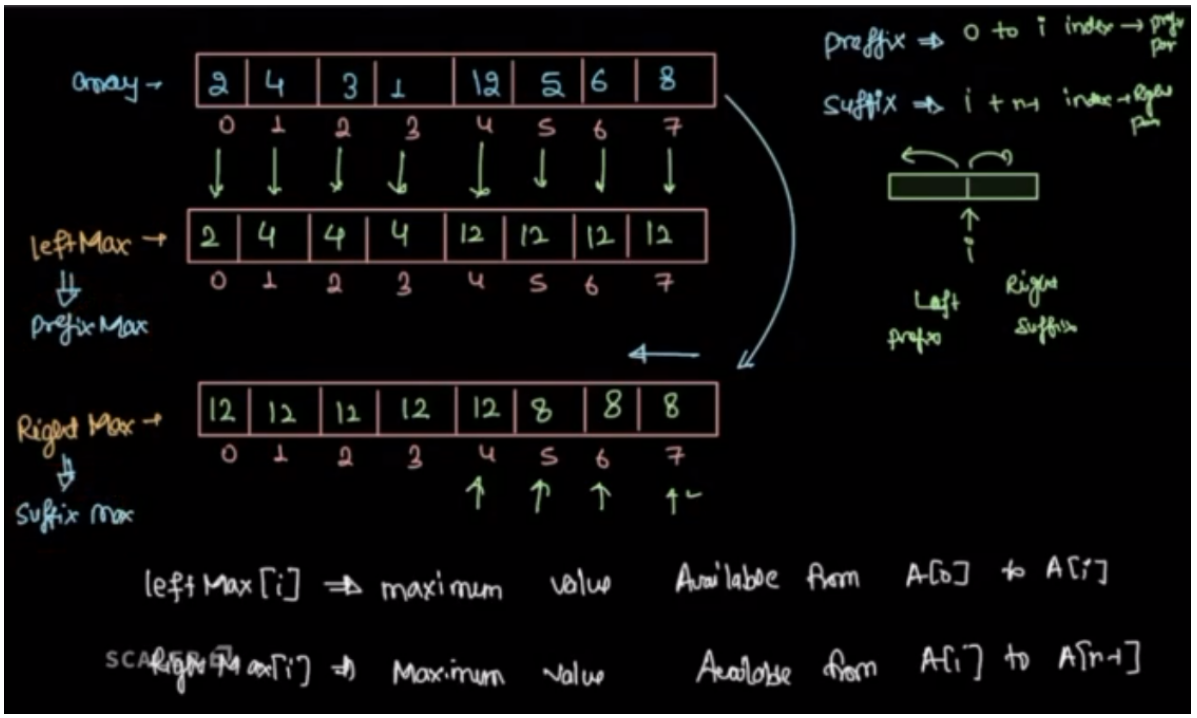
RainWater Trapping —----------

Kadane's Algorithm —----------

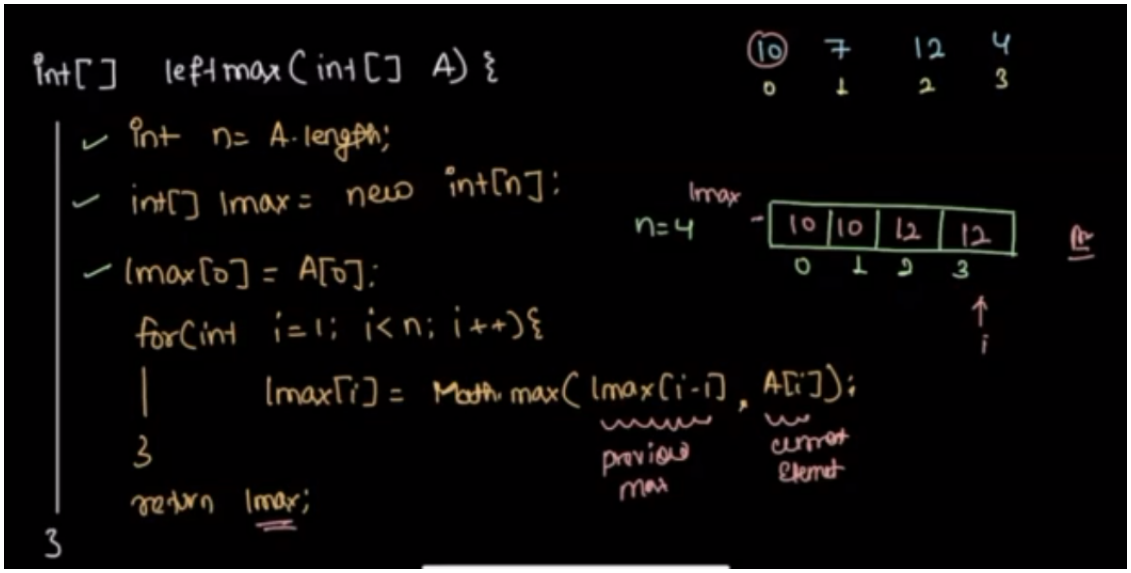
Leftmax and Rightmax —----------

Create PrefixMax (leftMax) and Suffix Max (rightMax)

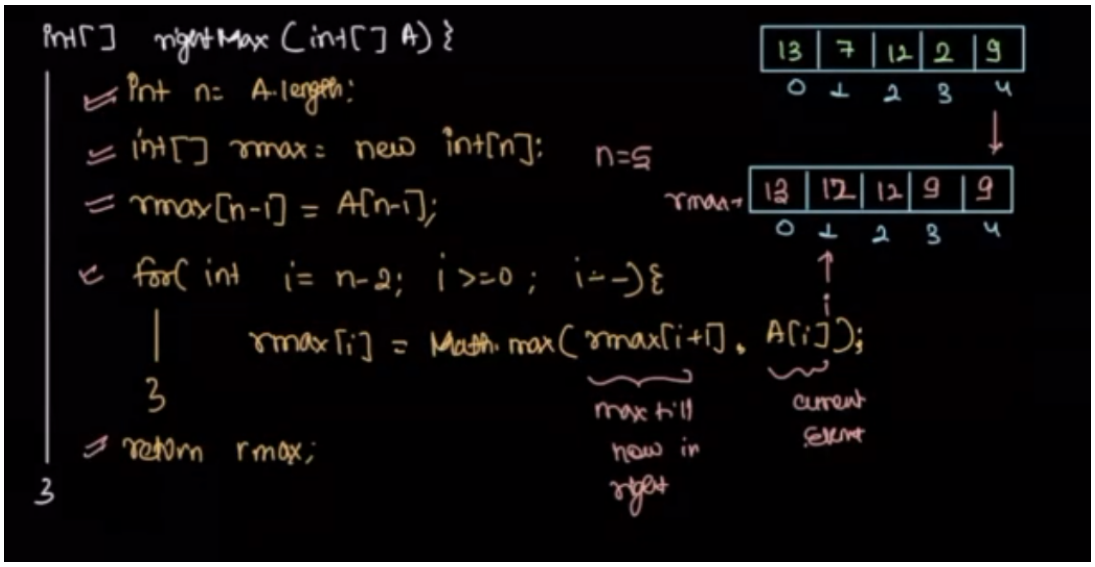




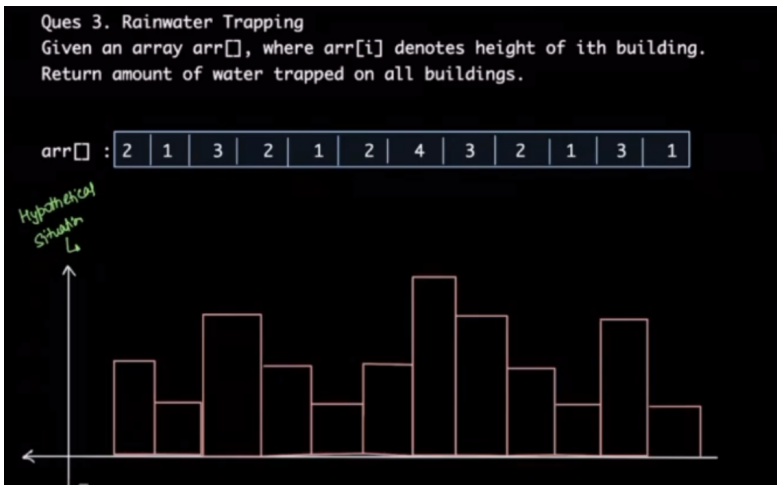
Left Max



Right Max - here we have writer n-2 second element from last because we are filling array form right same we did i+1 from right

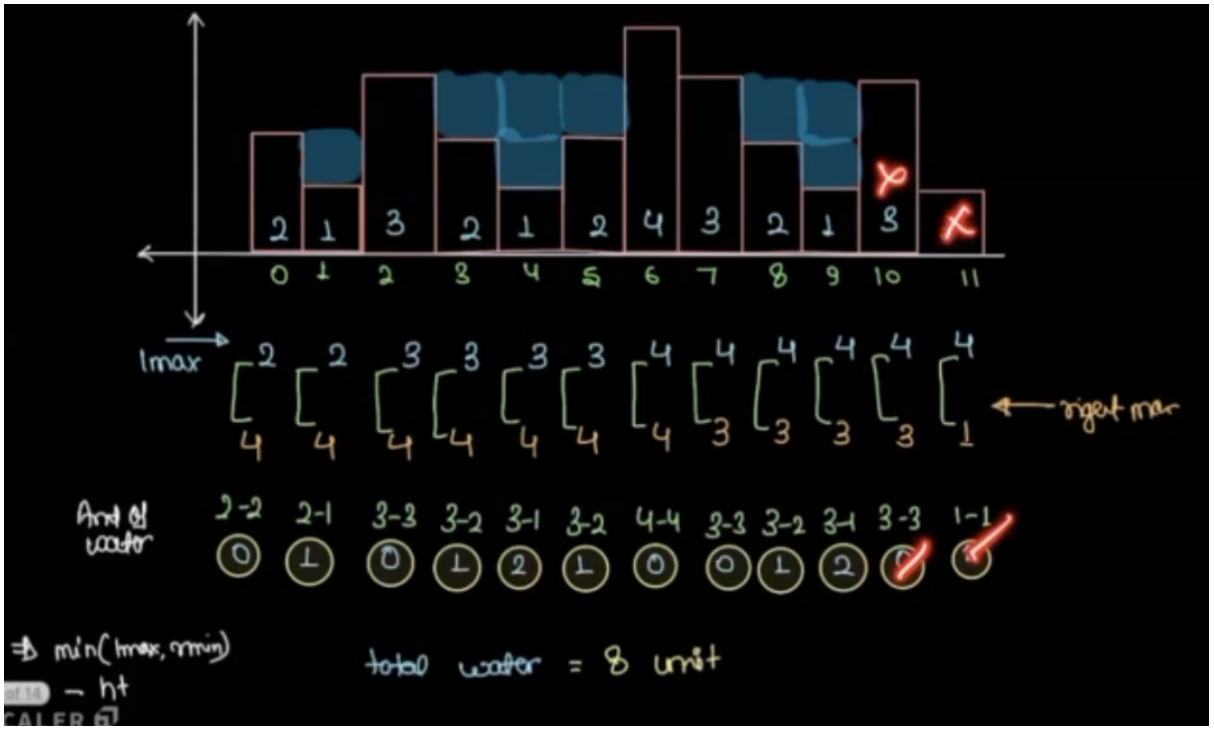


RainWater Trapping —----------



Count of water at ith building = Min(lmax, rmax) - height[i]

Note -> Want to calculate amount of water of ith building take help of leftMax[i] and rightMax[i]



Steps

1. Calcualte leftMax

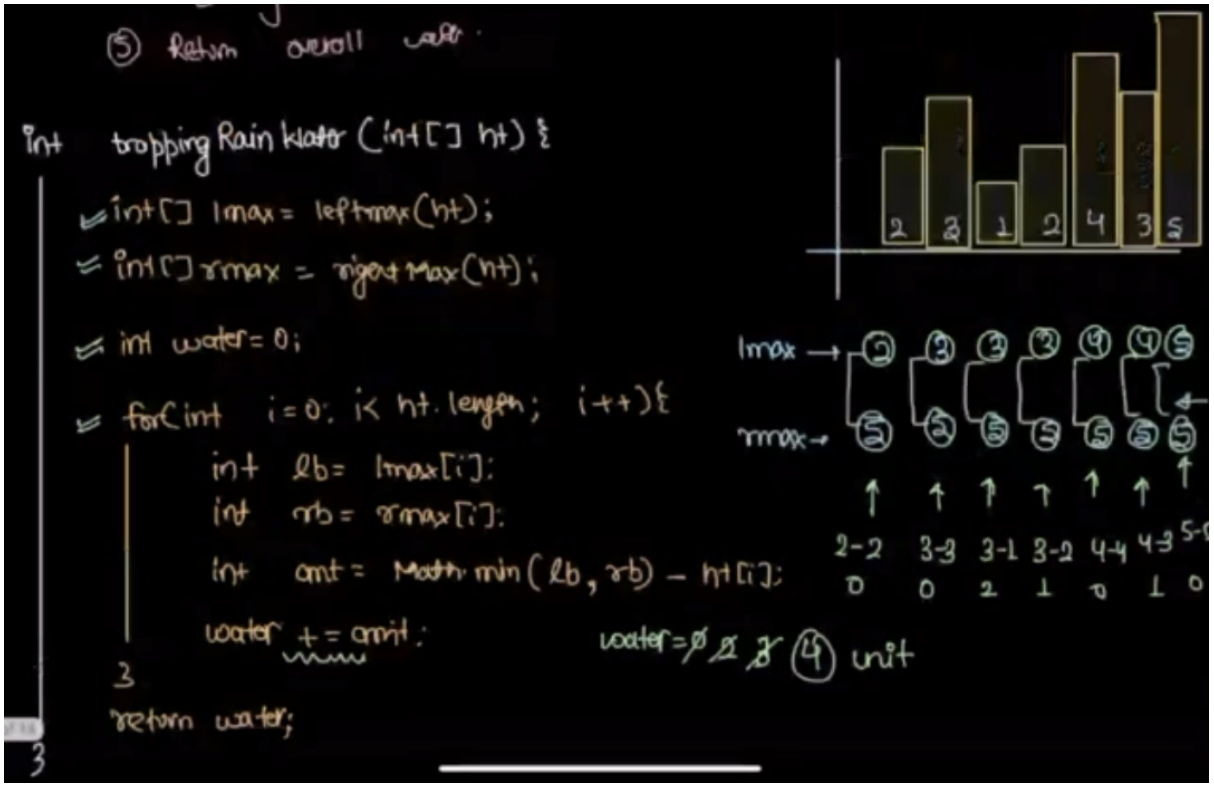
2. Calculate rightMax

3. Itrate from 0th to n-1th building

4. Calculate amount of water for the ith building & store it in overall water using

Count of water at ith building = Min(lmax, rmax) - height[i]

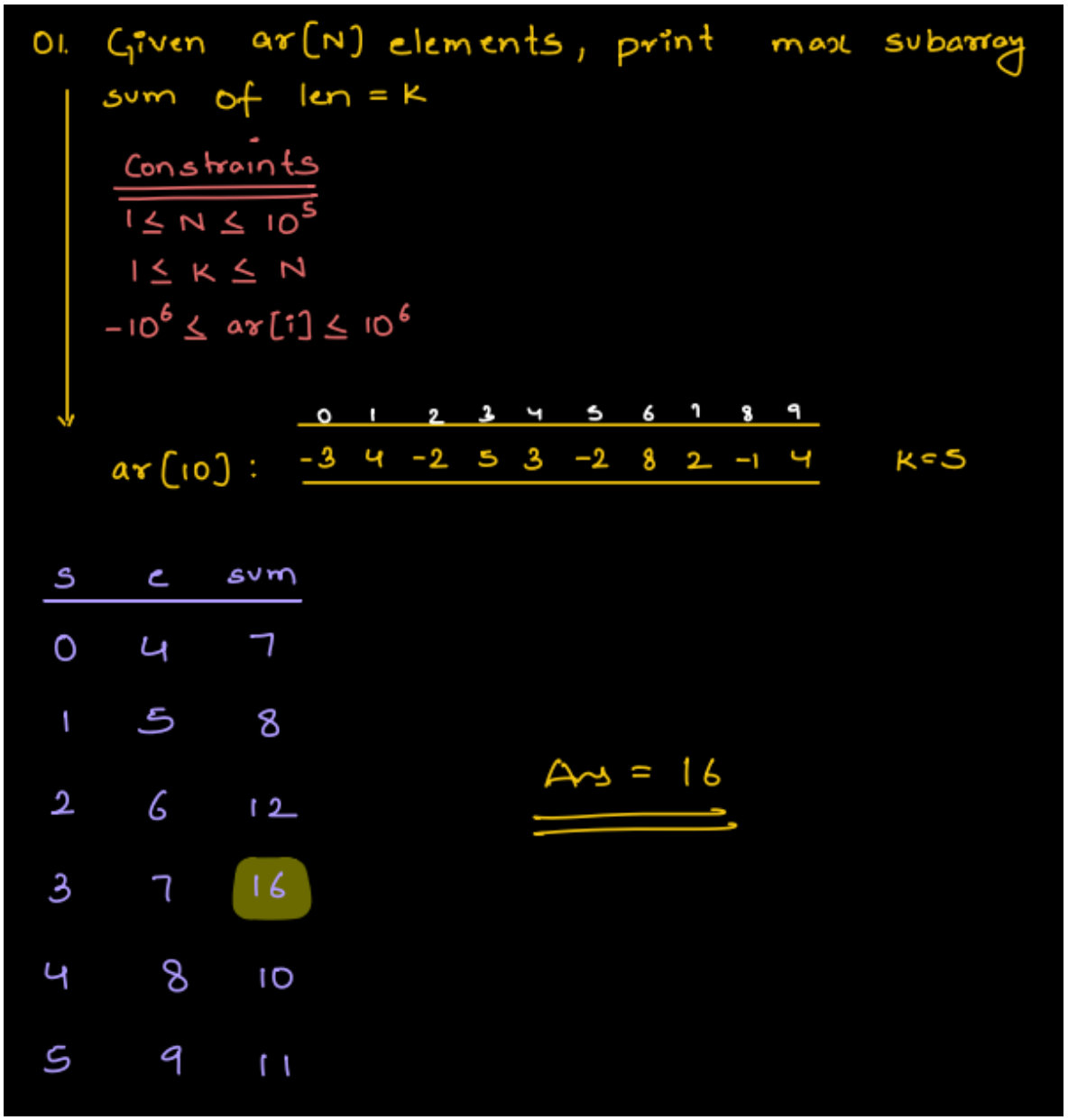
5. return overall water

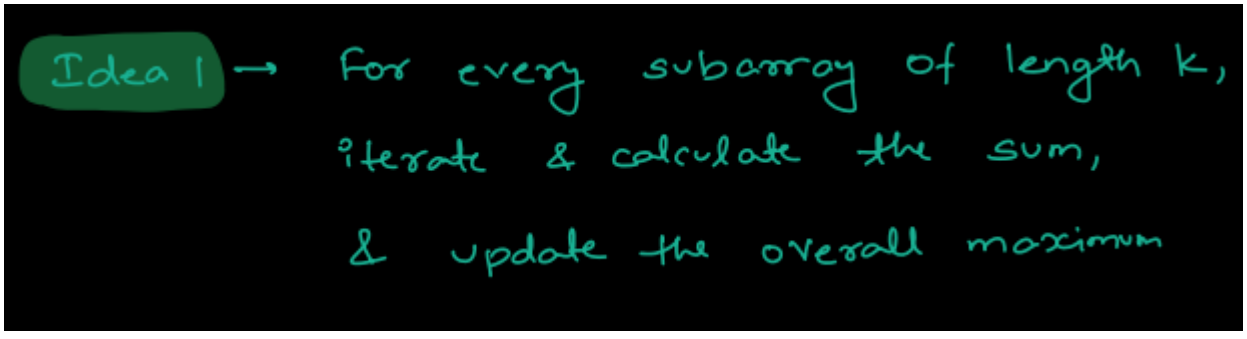


DSA: Arrays : 2D Matrices 1 - 6 - May 2023

- print max subarray of len=k
- sliding window approach
- print boundary in clockwise direction
- spiral printing
- search in rowwise sorted and coloumnwise sorted matrix

Print max subarray of len=k





```
int maxsubarrsum(int[] ar, int k)
    s=0 , e=k-1
    int ans = Integer.MIN_VALUE;
    while(e<n)
        int sum=0
        for(i=s ; i<=e ; i++){
            sum = sum + ar[i];
        }
        if(sum>ans) {ans=sum};
        s=s+1
        e=e+1
    }
    return ans
}
```

TC = No. of subarrays of length k * TC for each subarray

TC = $(N-K+1) * K$

K=1

TC = $(N-1+1)*1$

$= O(N)$

K = N/2

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

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

$= \frac{N^2}{4} + \frac{N}{2}$

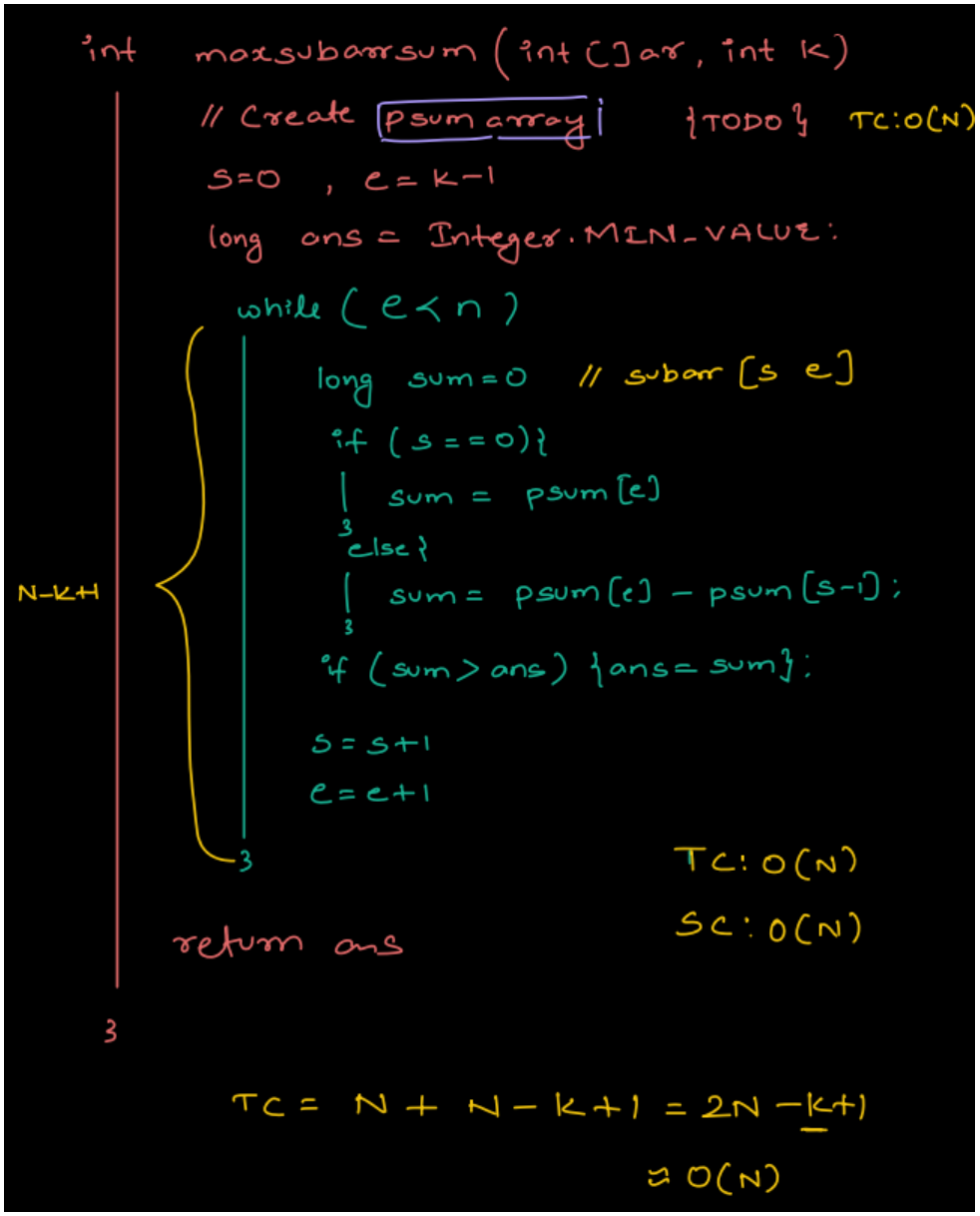
K = N

TC = $(N-N+1)N$

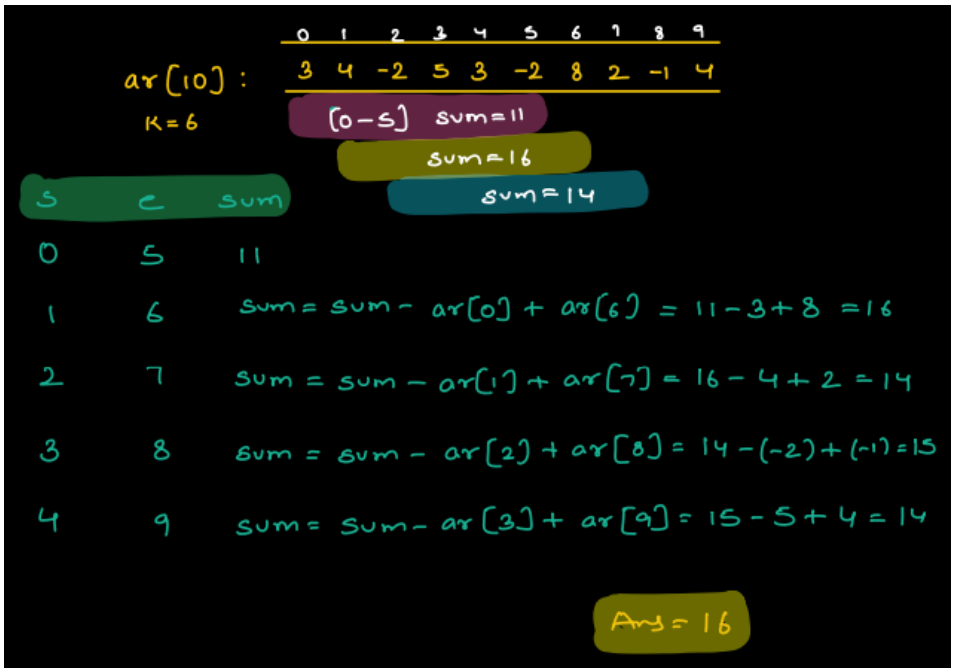
$= O(N)$

TC $\approx O(N^2)$

Idea 2 -> Get all the subarray sums of len = k using prefixsum technique



Idea 3 - Carry forward the ans for a fixed size window - Sliding window approach



```
int maxsubarrsum (int []ar, int k)
{
    long ans = -∞
    long sum = 0

    for (i=0; i<k; i++){
        sum = sum + ar[i];
    }                                   } k
    if (sum > ans) ans = sum ;

    s=1   e=k

    while (e<n)
    {
        sum = sum - ar[s-1] + ar[e]
        if (sum>ans){
            ans = sum
        }                               } n-k
        s=s+1
        e=e+1
    }
    return ans;
}
```

TC = K + n-K = O(n)

SC = O(1)

Note: Fixed size subaaray -> Sliding window - print boundary in clockwise direction

02 Given mat[N][N], print boundary in clockwise direction

Constraints

$1 \le N*N \le 10^6$

$-10^6 \le ar[i] \le 10^6$

Ex1 mat[5][5]

	0	1	2	3	4
0	1	2	3	4	5
1	16	17	18	19	6
2	15	24	25	20	7
3	14	23	22	21	8
4	13	12	11	10	9

Output = 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

Ex2 mat[3][3]

	0	1	2
0	1	2	3
1	8	9	4
2	7	6	5

Output = 1 2 3 4 5 6 7 8

Ex1 mat[5][5]

	0	1	2	3	4
0	1	2	3	4	5
1	16	17	18	19	6
2	15	24	25	20	7
3	14	23	22	21	8
4	13	12	11	10	9

Ideas = → n=5

01. Print 4 ele in 0th row = L→R
02. Print 4 ele in 4th col = T→D
03. Print 4 ele in 4th row = R→L
04. Print 4 ele in 0th col = D→T

mat[3][3]

	0	1	2
0	1	2	3
1	8	9	4
2	7	6	5

n=3

01. Print 2 ele in 0th row L→R
02. Print 2 ele in 2nd col T→D
03. Print 2 ele in 2nd row R→L
04. Print 2 ele in 0th col D→T

```
void printboundary (int [][] ar)
{
    int i=0 , j=0

    for(k=1 ; k<N ; k++){
        print (ar[i][j]);
        j=j+1
    }

    for(k=1 ; k<N ; k++){
        print (ar[i][j]);
        i=i+1
    }

    for(k=1 ; k<N ; k++){
        print (ar[i][j]);
        j=j-1
    }

    for(k=1 ; k<N ; k++){
        print (ar[i][j]);
        i=i-1;
    }

    if (n==1) { print (ar[i][j]) }   ⟶ Edge case
}
```

K	i	j	K<N
1	0	0	1<5
2	0	1	2<5
3	0	2	3<5
4	0	3	4<5
5	0	4	5<5

Constraints

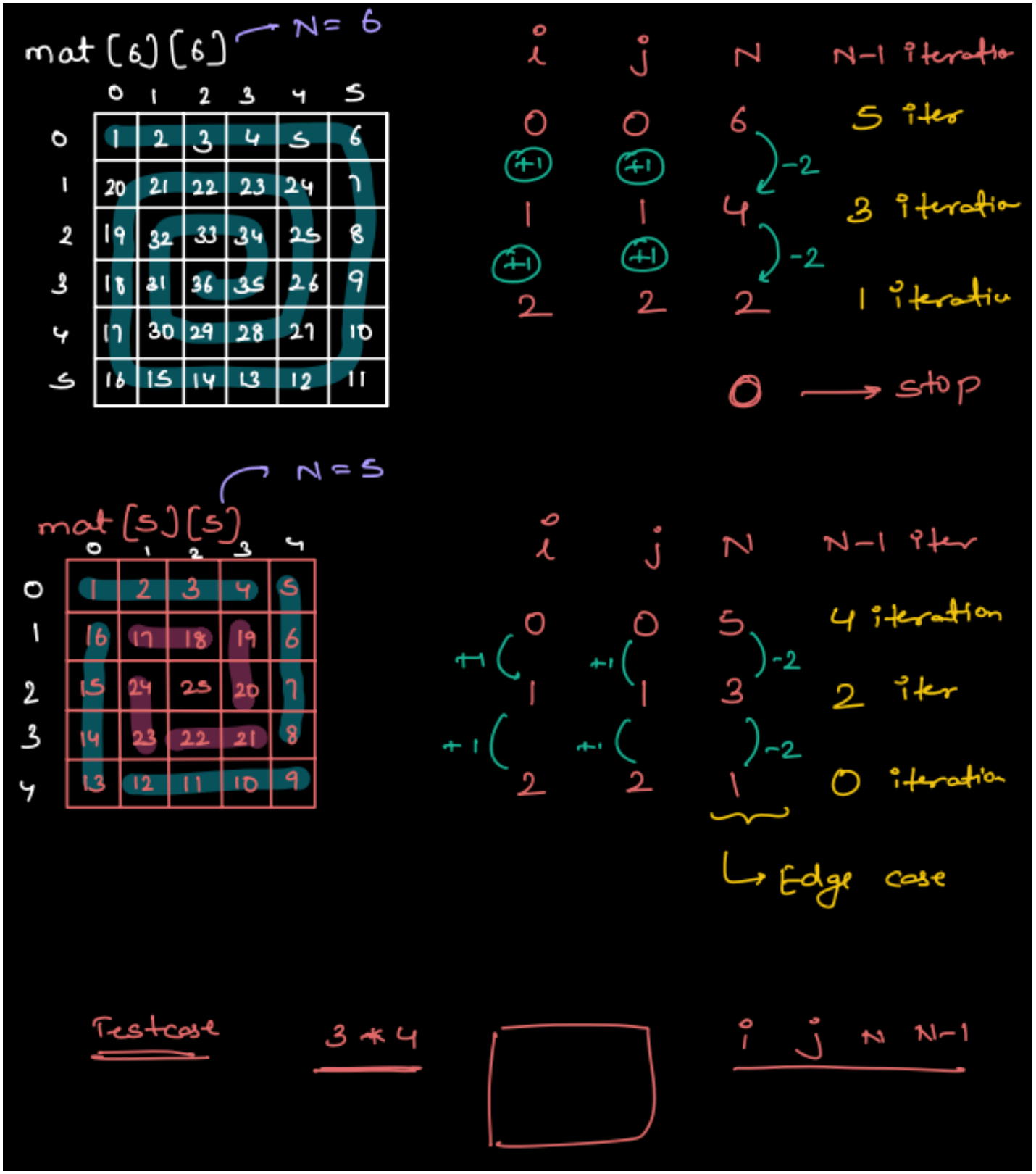
$1 \le N*N \le 10^6$

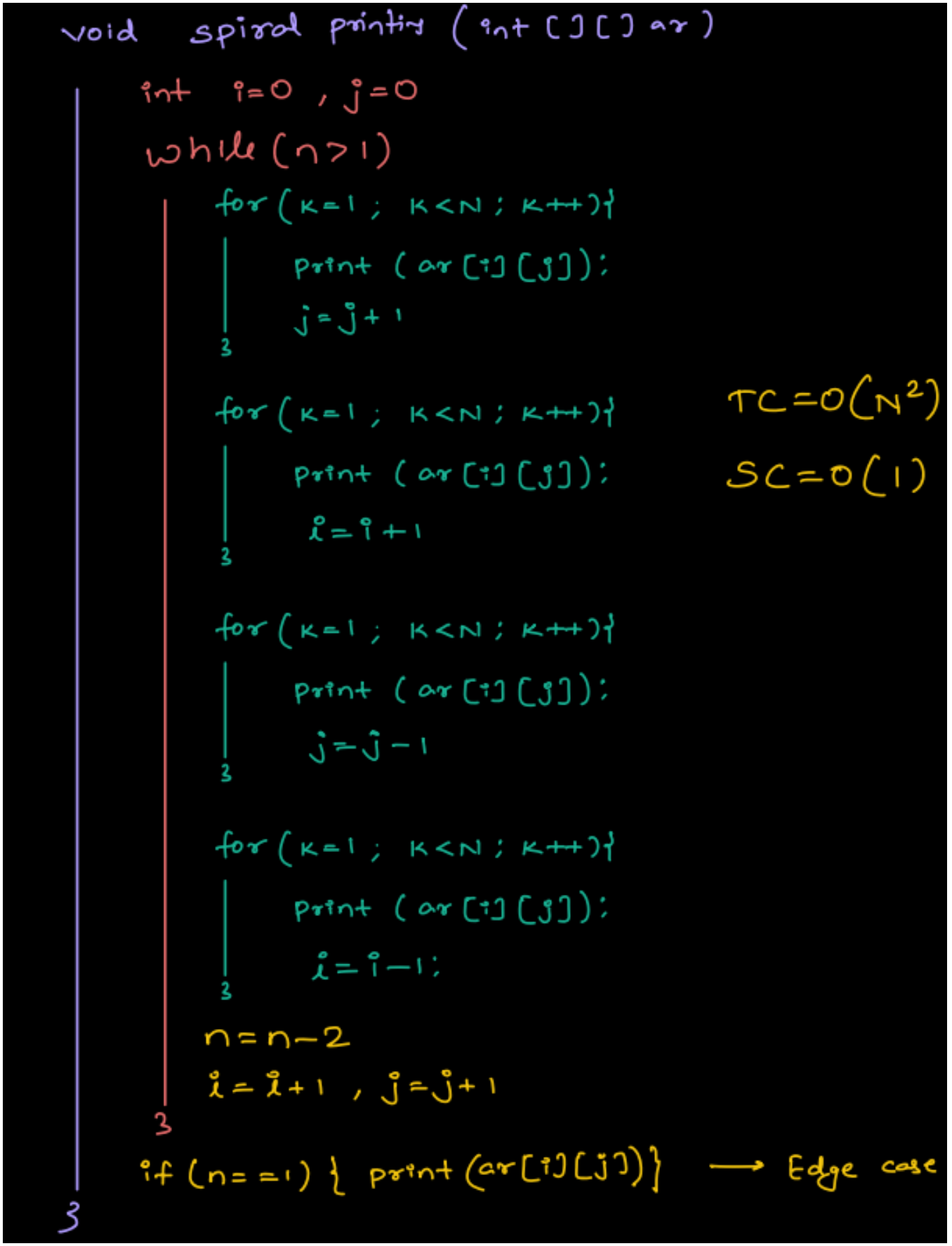
$-10^6 \le ar[i] \le 10^6$

TC: O(n)

SC: O(1)

spiral printing





Given rowwise & columnwise sorted matrix [N][M]. find k

Constraints

1 <= N * N <= 10^6

-10^6 <= mat[i][j] <= 10^6

Eg: mat[6][6] K=12 Ans = True

	0	1	2	3	4	5	
0	-10	-5	-2	2	4	7	<12
1	-7	-4	-1	3	6	9	<12
2	-2	3	5	7	11	14	>12
3	3	6	8	11	14	17	
4	7	11	12	15	19	20	
5	10	14	18	20	24	29	

Idea → Linear search on each & every row

TC : O(n*m)
SC : O(1)

→ return True

K=10

	0	1	2	3	4	5	
0	-10	-5	-2	2	4	7	<10
1	-7	-4	-1	3	6	9	9<10
2	-2	3	5	7	11	14	14>10 11>10 7<10
3	3	6	8	11	14	17	11>10 8<10
4	7	11	12	15	19	20	12>10 11>10 7<10
5	10	14	18	20	24	29	10==10 → Ans = True

```
boolean matrixSearch(int[][] arr, int n, int m)
    i=0, j=m-1   // i=n-1, j=0 {TODO}
    while(i<n && j>=0)
        if(arr[i][j]==k)
            return true;
        else if(arr[i][j]>k)
            j=j-1
        else if(arr[i][j]<k)
            i=i+1;
    return false;
```

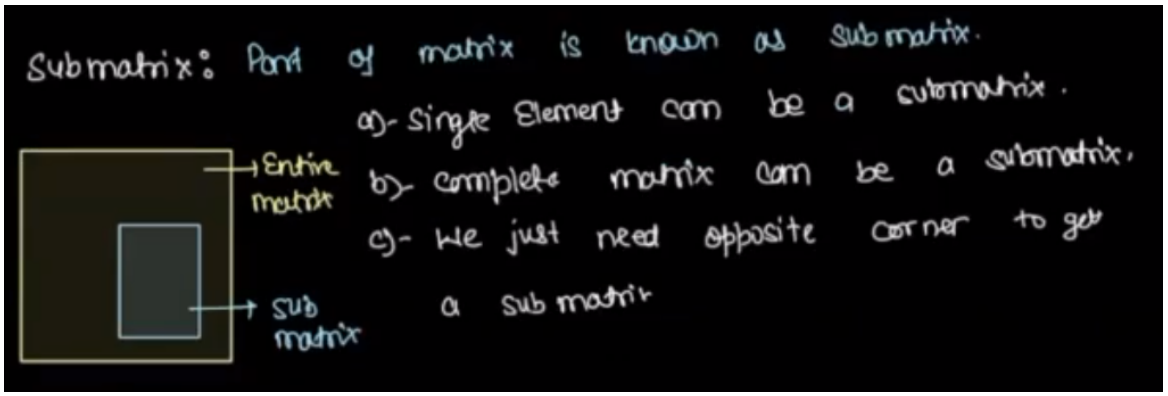
TC: O(n+m)
SC: O(1)

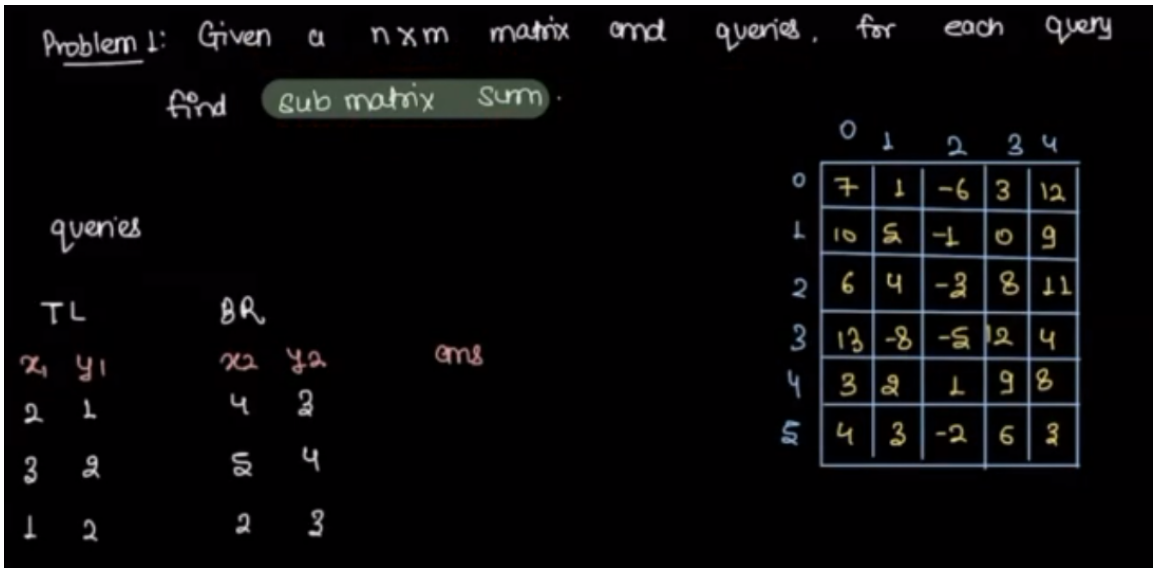
DSA: Arrays : 2D Matrices 2 - 9 - May 2023

Agenda ->

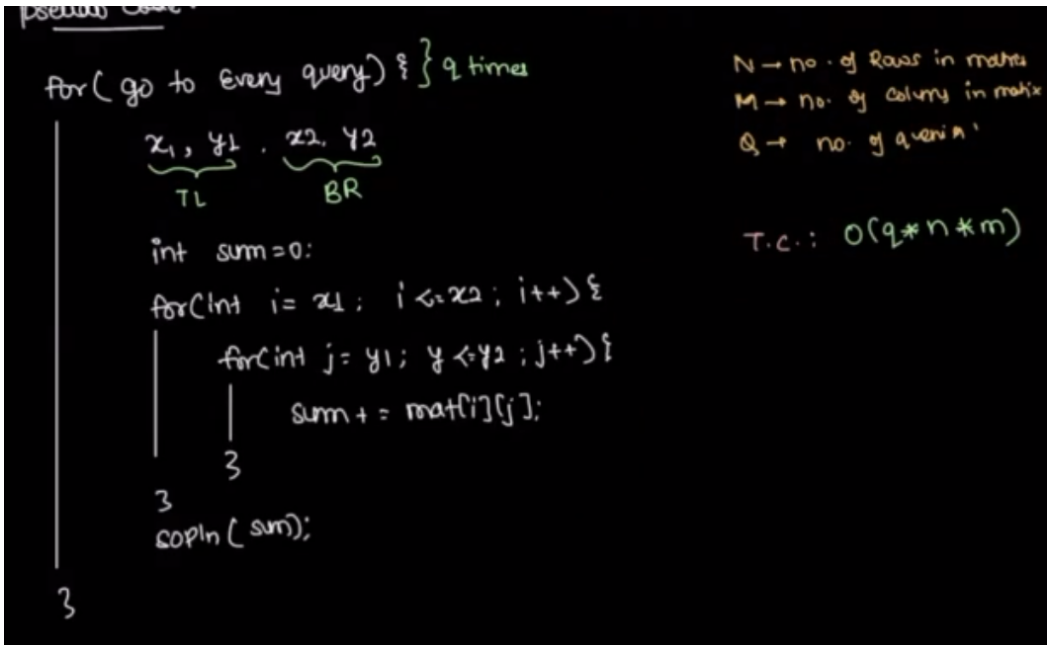
1. Submatrix sum queries
2. Max submatrix sum
3. Sum of all submatrices

What is Submatrix





Brute force -> Pseudo code



ii)- Optimised : prefix Sum

1D array at i^{th} index

psum[i] => Sum of all Elements from 0 to i. index.

2D array at (i,j) index

psum[i][j] => Sum of all Elements in submatrix having TL = (0,0) & B.R is (i,j)

A:

	0	1	2	3
0	3	2	4	1
1	-1	4	3	2
2	2	7	6	3

psum →

	0	1	2	3
0	3	5	9	10
1	2	8	15	18
2	4	17	30	36

How to generate prefix sum matrix using given matrix:

	0	1	2	3
0	3	2	4	1
1	-1	4	3	2
2	2	7	6	3

1. Row wise prefix sum →

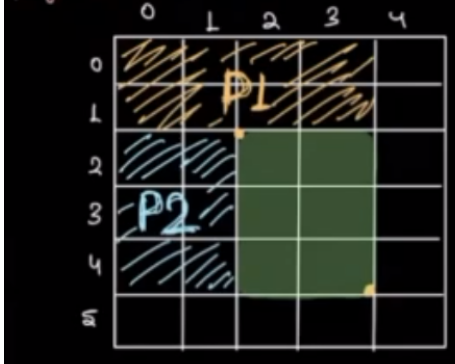
	0	1	2	3
0	3	5	9	10
1	-1	3	6	8
2	2	9	15	18

2. Column wise prefix Sum ↓

	0	1	2	3
0	3	5	9	10
1	2	8	15	18
2	4	17	30	36

If we have 2D prefix Sum matrix, How to solve any query ??

prefix Sum matrix: →



TL (2,2) BR (4,3)

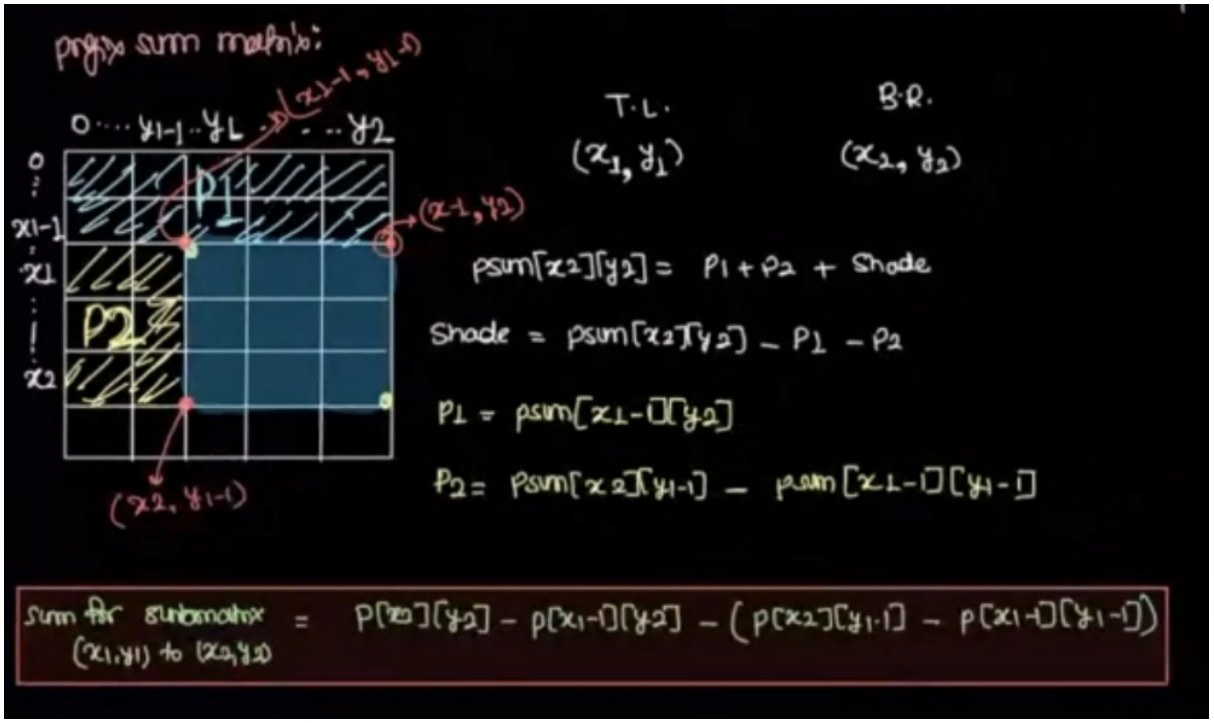
psum[4,3] = P1 + P2 + Shade Region

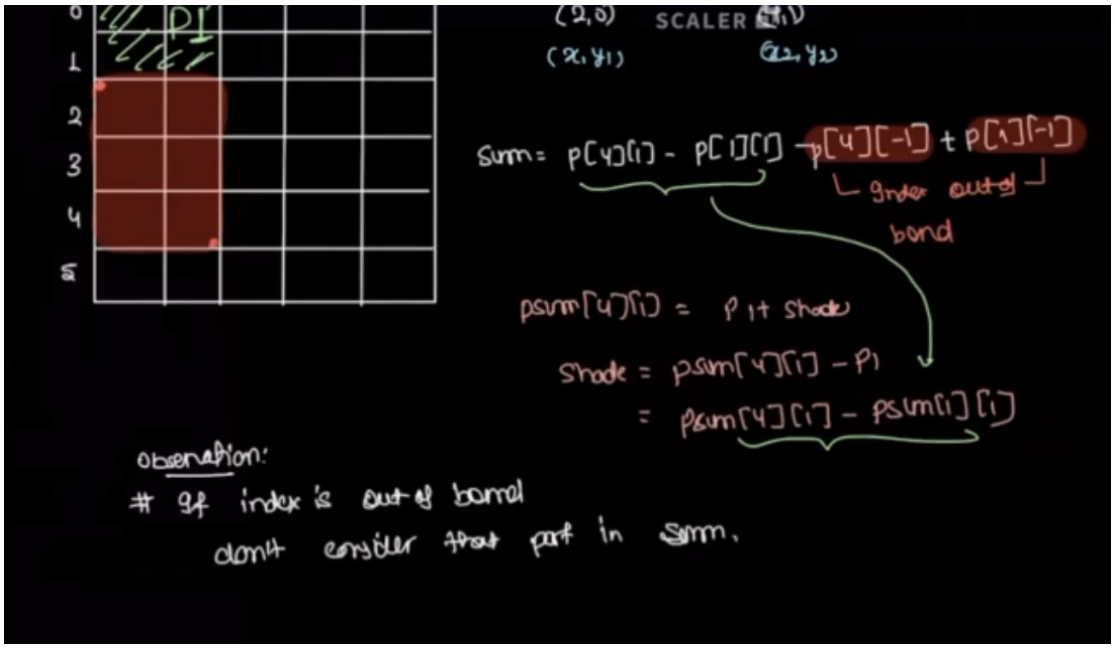
Shade Region : psum[4][3] − Part1 − Part2

Part1 = psum[1][3]

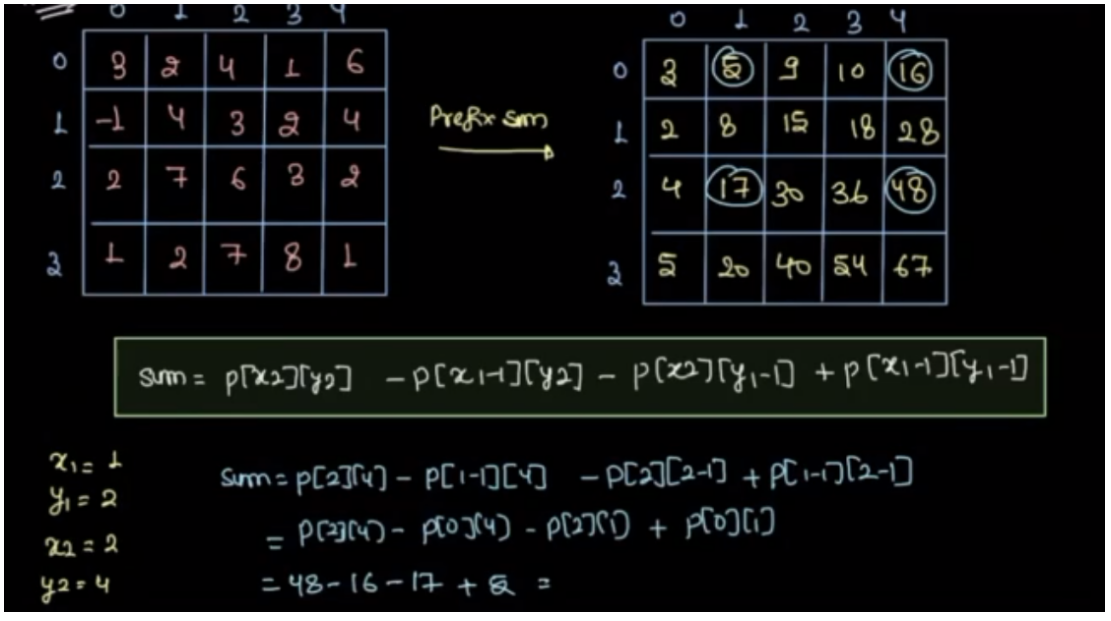
Part2 = psum[4][1] − psum[1][1]

Shaded Region = psum[4][3] − (psum[1][3]) − (psum[4][3] − psum[1][1])





If index is in minus then we have to apply condition x1 >0 and y1 > 0



```
void solve ( int[][] A,  int[][] query ){
    int[][] psum = prefixSum2D ( A );
    for(int q=0; q < query.length; q++){
        int x1 = query[q][0];
        int y1 = query[q][1];
        int x2 = query[q][2];
        int y2 = query[q][3];
        // x1 & y1 → TL   &   x2 & y2 → BR, find & print sum.
        int sum = psum[x2][y2];
        if(x1 > 0){
            sum -= psum[x1-1][y2];
        }
        if(y1 > 0){
            sum -= psum[x2][y1-1];
        }
        if(x1 > 0 && y1 > 0){
            sum += psum[x1-1][y1-1];
        }
        SOPln(sum);
    }
}
```

A:

	0	1	2	3
0	2	2	4	2
1	1	0	3	2

psum →

	0	1	2	3
0	2	4	8	11
1	3	5	12	17

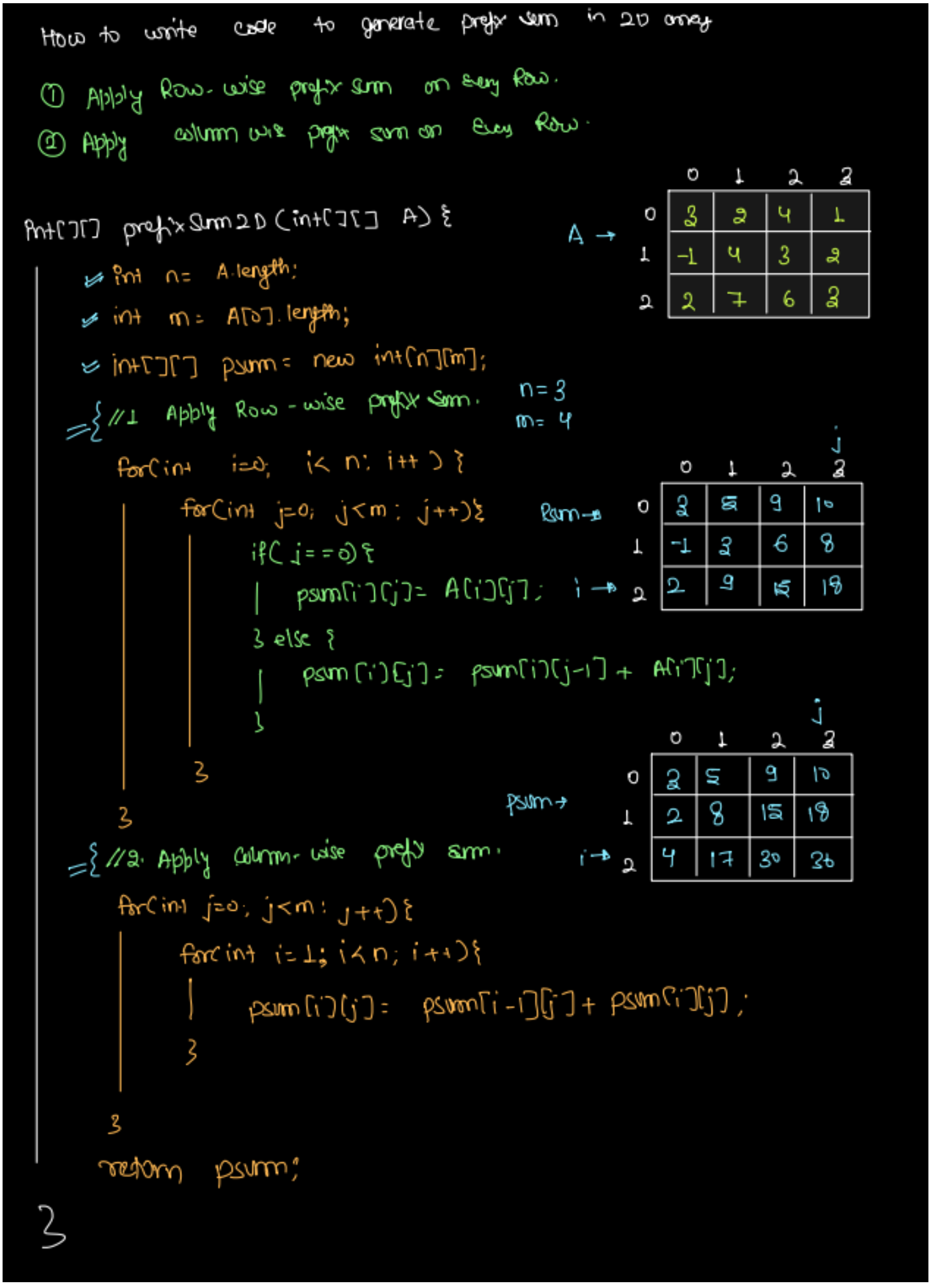
q { TL → (0,1)
BR → (1,3)

$x_1 = 0$
$y_1 = 1$ →
$x_2 = 1$
$y_2 = 3$

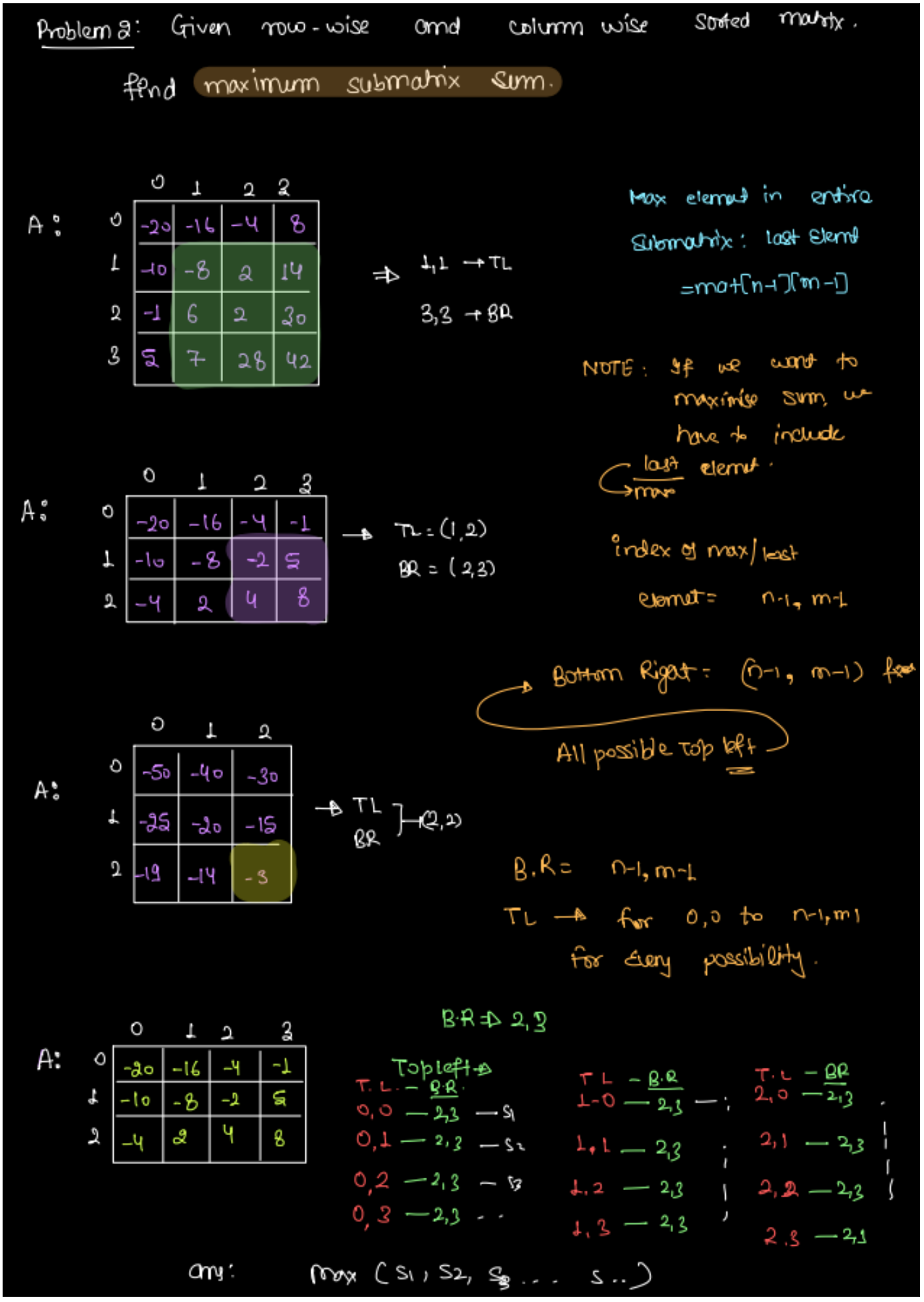
sum = 17 − psum[1][0]
= 17 − 3 = 14

→ o/p: 14

Prfix sum subMetrix



Problem -2 - find maximum submatrix sum



If we have prefix sum 2D array,

T.L. index & B.R index

T.C. for calculation of sum = O(1)

just Apply formula:

$$sum = p[x_2][y_2] - p[x_1-1][y_2] - p[x_2][y_1-1] + p[x_1-1][y_1-1]$$

Steps for solution:

1. Generate prefix sum for 2D array.
2. B.R is fixed ⇒ n-1, m-1
3. Iterate on all possible T.L. indexes
4. Calculate sum for TL & BR using psum & maximize sum.

```
int solve(int[][] A){
    int[][] psum = prefixSum2D(A);   ] ⇒ O(n*m)
    int max = Integer.MIN_VALUE;
    int n = A.length;
    int m = A[0].length;
    // BR index is fixed & it is n-1, m-1
    int x2 = n-1;
    int y2 = m-1;
    for(int x1 = 0; x1 < n; x1++){
        for(int y1 = 0; y1 < m; y1++){
            // x1 & y1 => TL & x2 & y2 => BR
            int sum = psum[x2][y2];
            if(x1 > 0){
                sum -= psum[x1-1][y2];
            }
            if(y1 > 0){
                sum -= psum[x2][y1-1];
            }
            if(x1 > 0 && y1 > 0){
                sum += psum[x1-1][y1-1];
            }
            max = Math.max(max, sum);
        }
    }
    return max;
}
```

O(n*m)

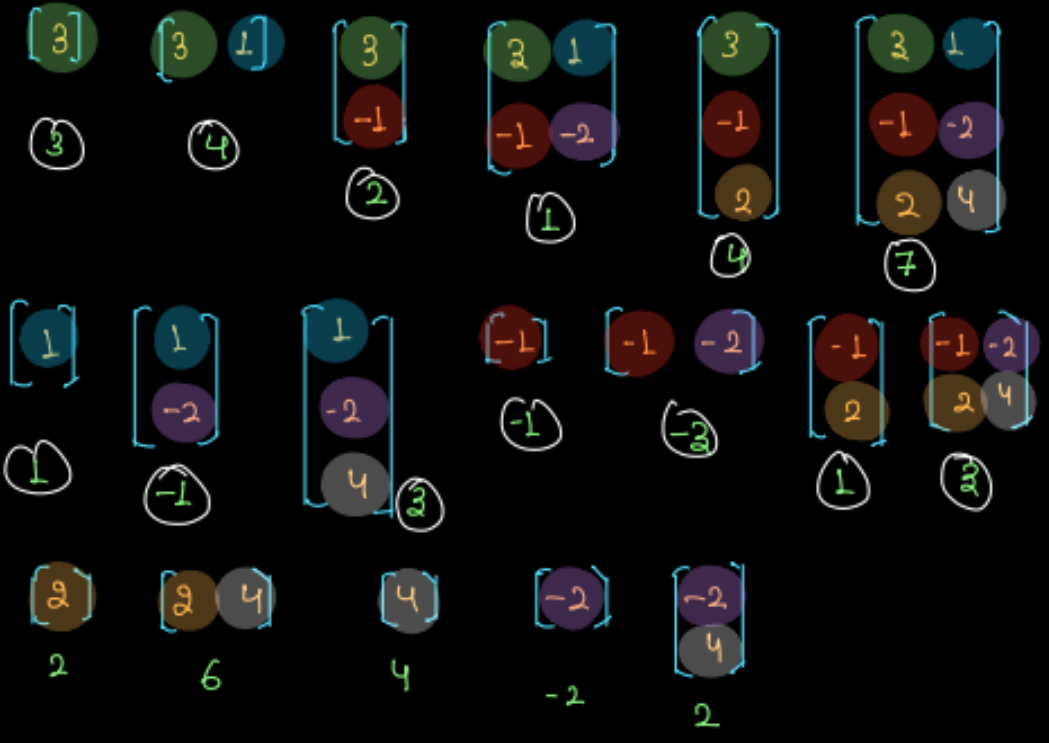
T.C. : O(n*m)

S.C : O(n*m)

Problem 2: Given a n*m matrix, find sum of all submatrices sum.

A:

	0	1
0	3	1
1	-1	-2
2	2	4



sum = 3+4+2+1+4+7+~~X~~+(-1)+~~2~~+(-1)+(-3)+~~X~~+3+2+6+4+(-2)~~*2~~

= 36 Ans

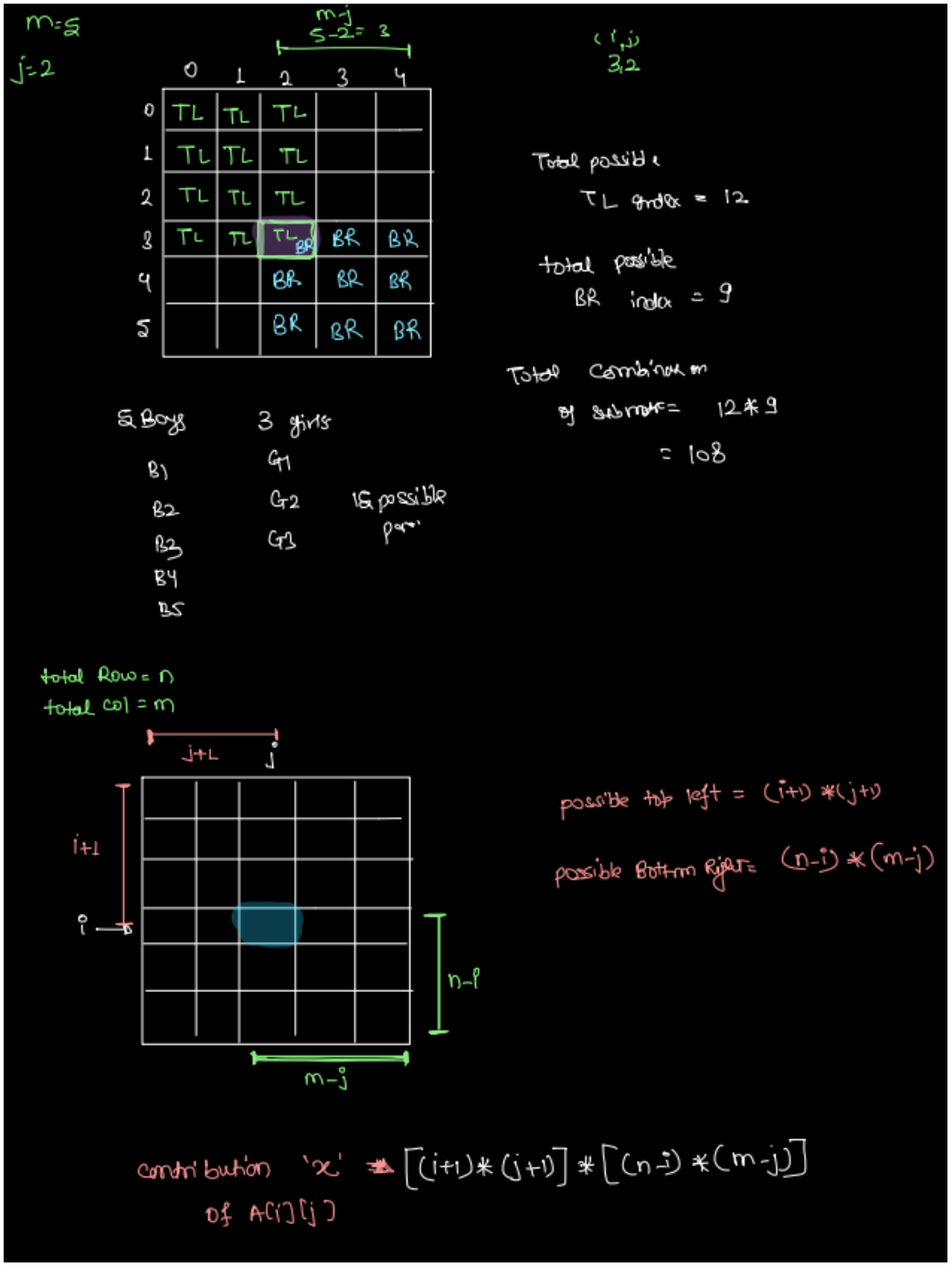
A[0][0] A[0][1] A[1][0] A[1][1] A[2][0] A[2][1]

Sum = 6*3 + 6*1 + 8*(-1) + 8*(-2) + 6*2 + 6*4

= ~~18~~ + ~~6~~ - ~~8~~ ~~-16~~ + 12 + 24

= 36 Ans

Conclusion: if A[i][j] is coming x-times in entire sum of all Submatrix, the contribution of A[i][j] in sum is x * A[i][j]




```
int solve(int[][] A) {
    int n = A.length;
    int m = A[0].length;
    int ans = 0;
    for(int i=0; i<n; i++){
        for(int j=0; j<m; j++){
            int contribution = (i+1)*(j+1)*(n-i)*(m-j);
            ans += contribution * A[i][j];
        }
    }
    return ans;
}
```

T.C: O(n*m)
S.C = O(1)

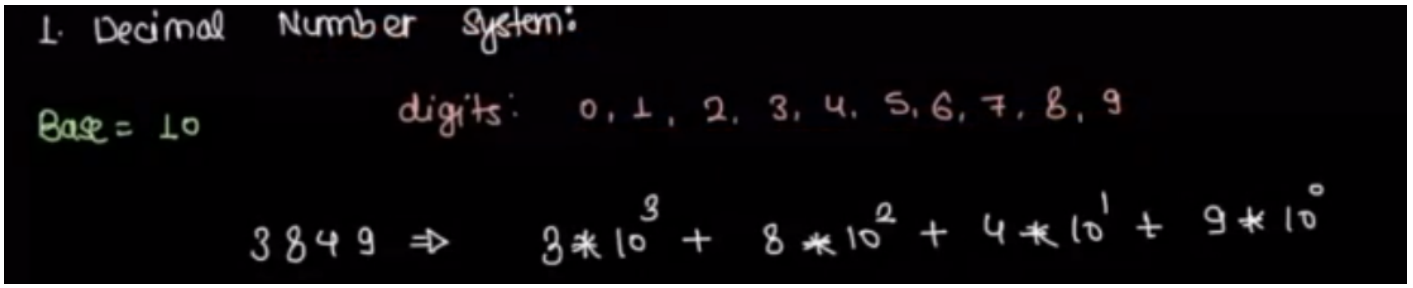
SCALER

DSA: Bit Manipulation -1 - 11 - May 2023

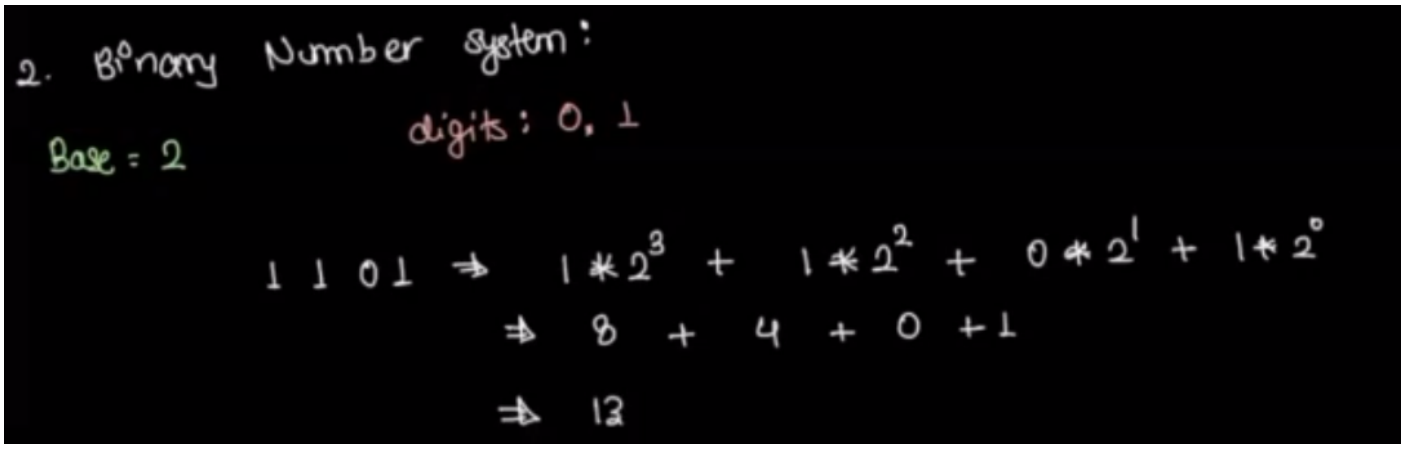
- **Number System Decimal + Binary—----------**
- **Addition of number (dec + binary)—----------**
- **Bitwise Operator—----------**
- **Commutative property—----------**
- **Prob : Single Element—----------**
- **Left Shift—----------**
- **Right Shift—----------**

Number System Decimal + Binary

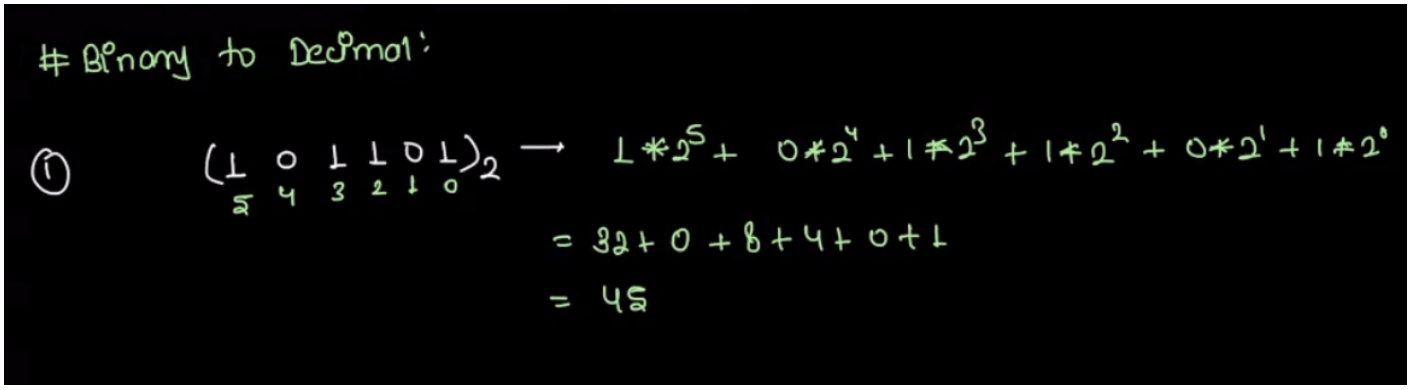
1. Decimal Number System -> Base 10 Digits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9



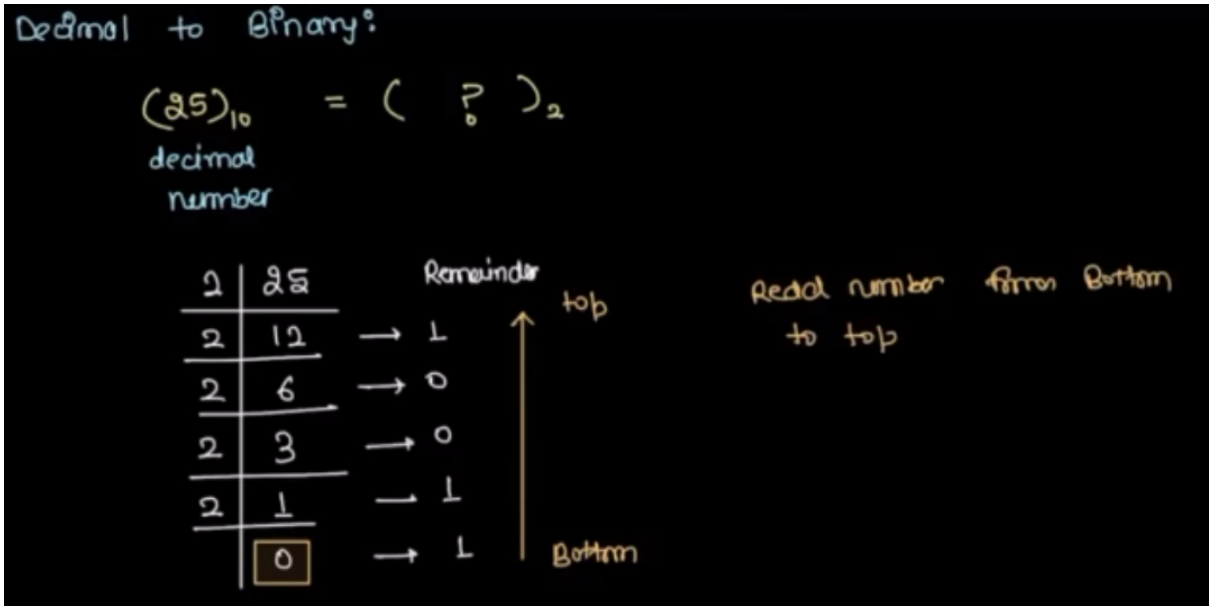
2. Binary Number System



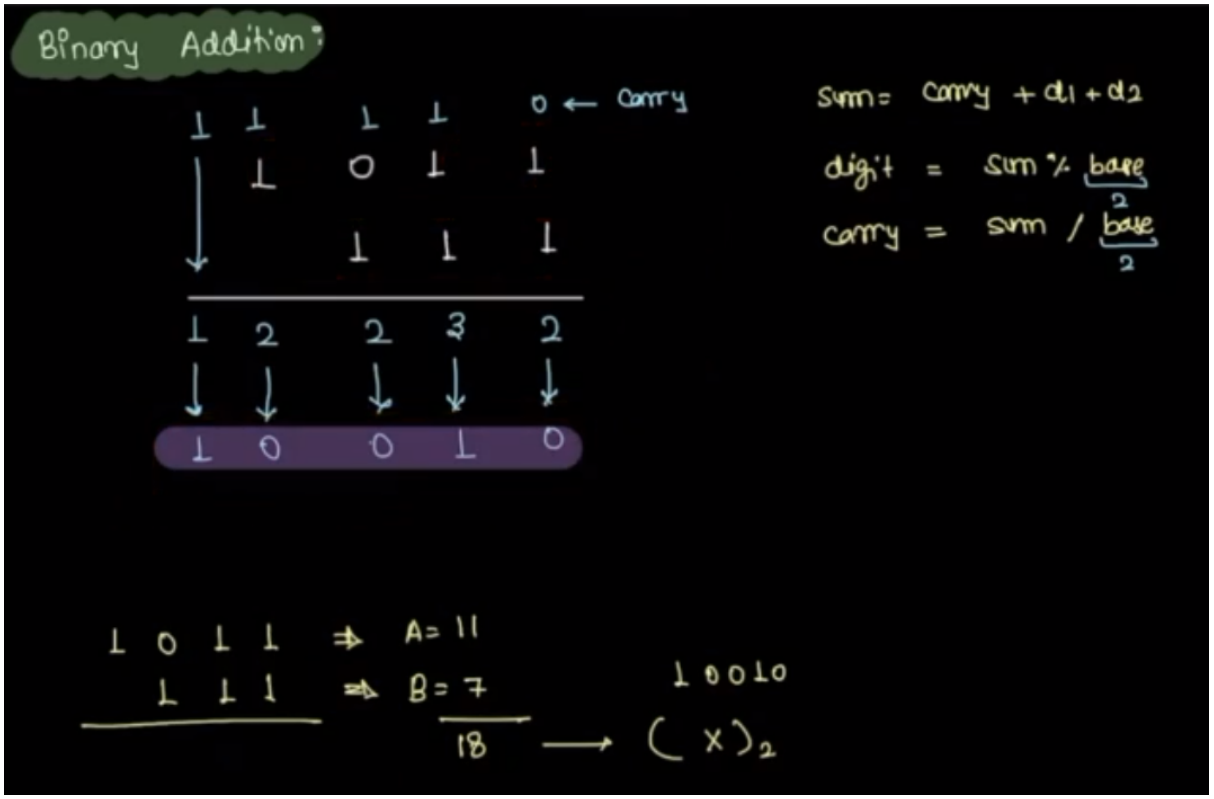
Use index for power

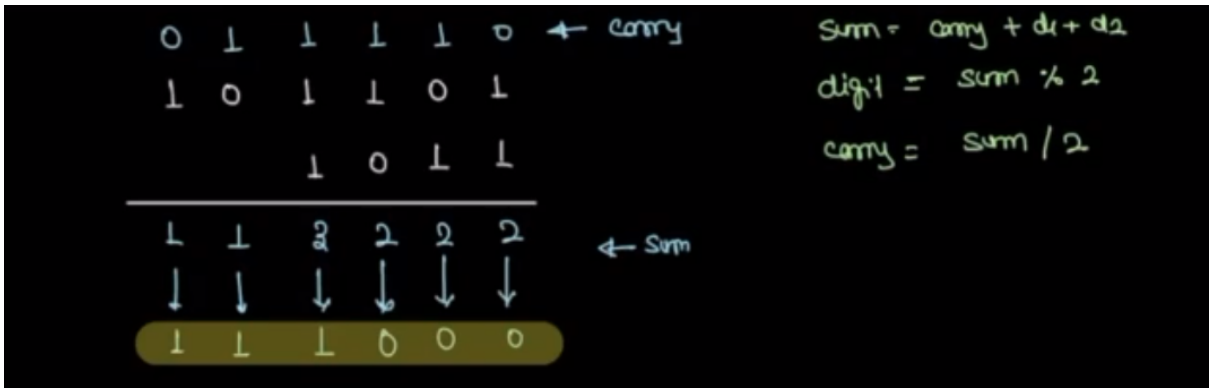


Decimal to Binary



Addition of Binary Numbers

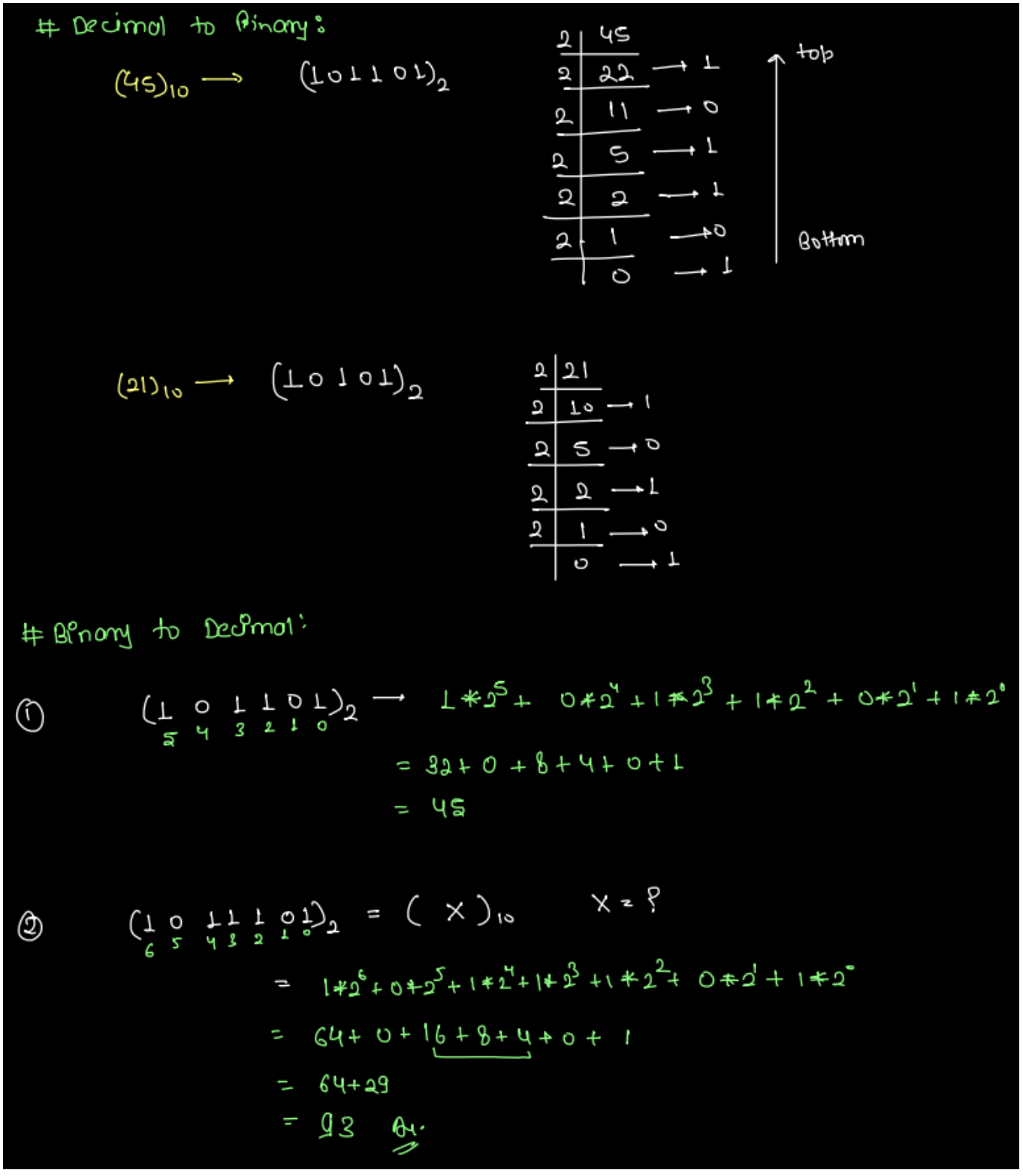




DSA: Bit Manipulation - 2 - 13 - May 2023

- Quick recap
- Check Ith Bit
- Count set bit
- Power of left shift
- Negative Number
- Range of number

Quick recap



Bitwise operator:

`&` , `|` , `^` , `~` , `<<` , `>>`

AND OR XOR NOT left shift Right shift

1 | 0 ⇒ 1
1 | 1 ⇒ 1
} 1 is dominant

0 & 0 = 0
1 & 0 = 0
} 0 is dominant
1 & 1 = 1

AND

```
Sopln( 13 & 10);
```

↳ o/p: 8

13 → 1 1 0 1
10 → 1 0 1 0
& → 1 0 0 0 = 8

NOTE: 0 is dominant in AND

OR

```
Sopln (13 | 10):
```

↳ o/p: 15

13 → 1 1 0 1
10 → 1 0 1 0
| → 1 1 1 1 → 15

NOTE: 1 is dominant in OR

XOR

```
Sopln(13 ^ 10) :
```

↳ 7

13 → 1 1 0 1
10 → 1 0 1 0
^ → 0 1 1 1 → 7

NOTE: Same Same puppy Shame

same same ⇒ 0
otherwise → 1

Left Shift:

Reload if not

SOPln(15 << 2)

$a << n = a * 2^n$

$a << n$

$15 << 2 \rightarrow$ 1 1 1 1

$15 * 2^2$

$= 15 * 4 = 60$

$[1\ 1\ 1\ 1\ 0\ 0]_2$ (positions 5 4 3 2 1 0)

0 inserted from Right

bits shifted towards left

$15 \rightarrow 8+4+2+1$

$\hookrightarrow (1\ 1\ 1\ 1)_2$

$1*2^5+1*2^4+1*2^3+1*2^2+0*2^1+0*2^0$

$32+16+8+4+0$

$= 40+20+0$

$= 60$

Right Shift:

SOPln(200 >> 4);

↳ op: 12

$a >> n = \frac{a}{2^n}$

$\frac{200}{2^4} = \frac{200}{16} = 12$

NOTE: Every operation in bitwise will work in O(1) time complexity.

$a = 10 = 8+2$

$= 01010$

$1 \rightarrow 00001$

$a \wedge 1 \Rightarrow$ 0 1 0 1 0 / 0 0 0 0 1

1 0 1 1

Right shift by 2

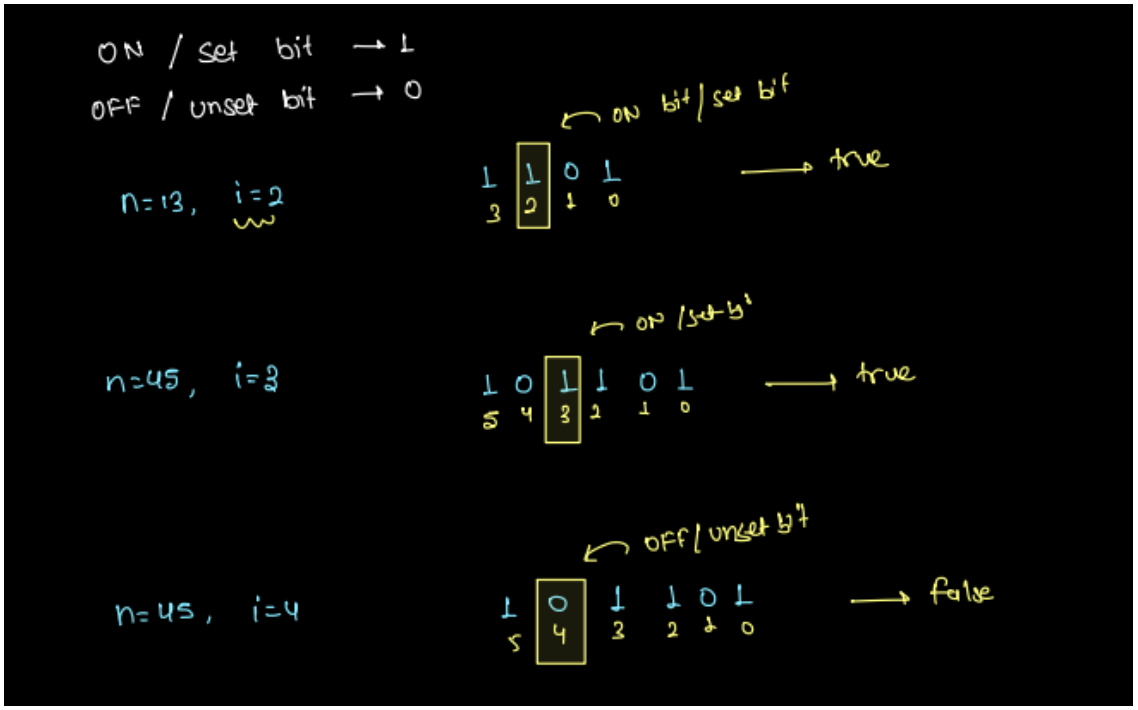
Initially → $(1\ 0\ 1\ 0\ 1\ 0\ 0)_2$

After shift → $(1\ 0\ 1\ 0\ 1\ 1)_2$ discarded

In Right shift:

$a >> n \Rightarrow \frac{a}{2^n}$

Problem: Given an integer number 'n' & index 'i' and if ith bit is on(i.e. set) or Off(ie, unset)

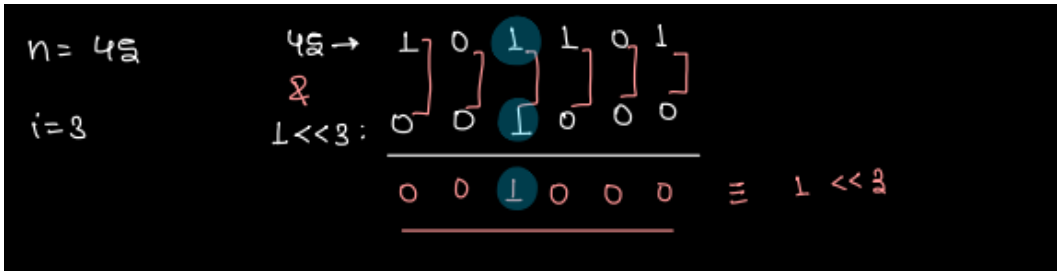


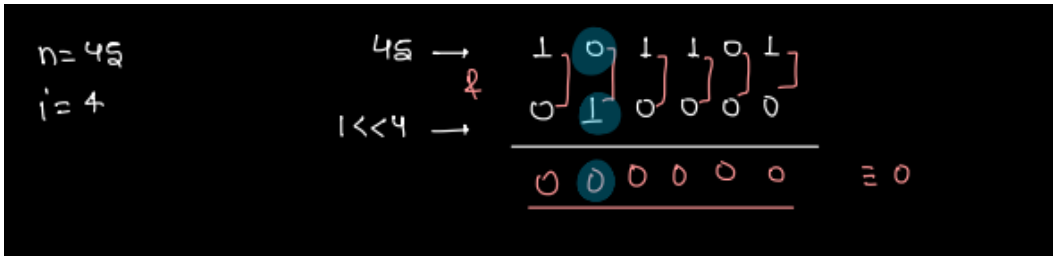
Bruteforce -------

Idia1: Convert n into binary and check ith bit whether it is on or off

Todo

Idea2: using AND

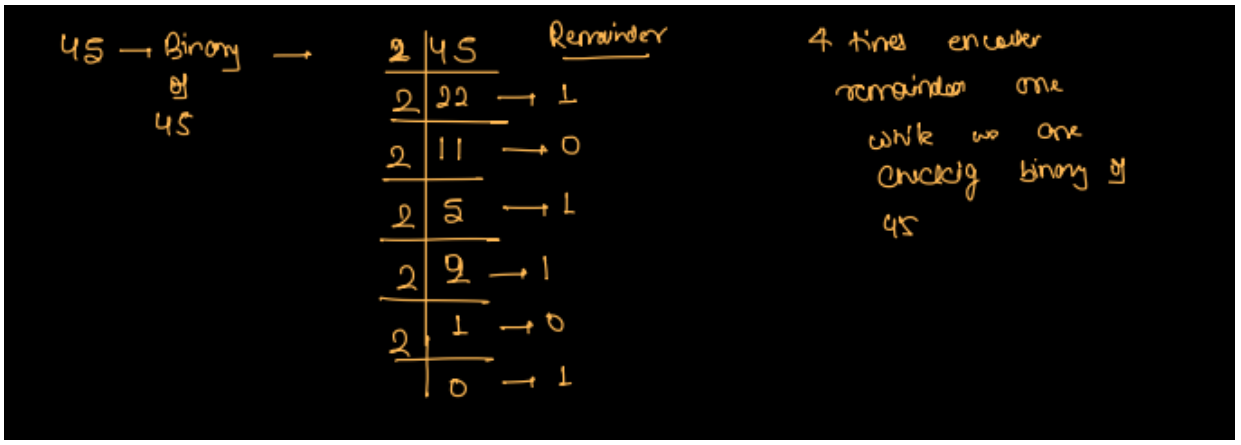


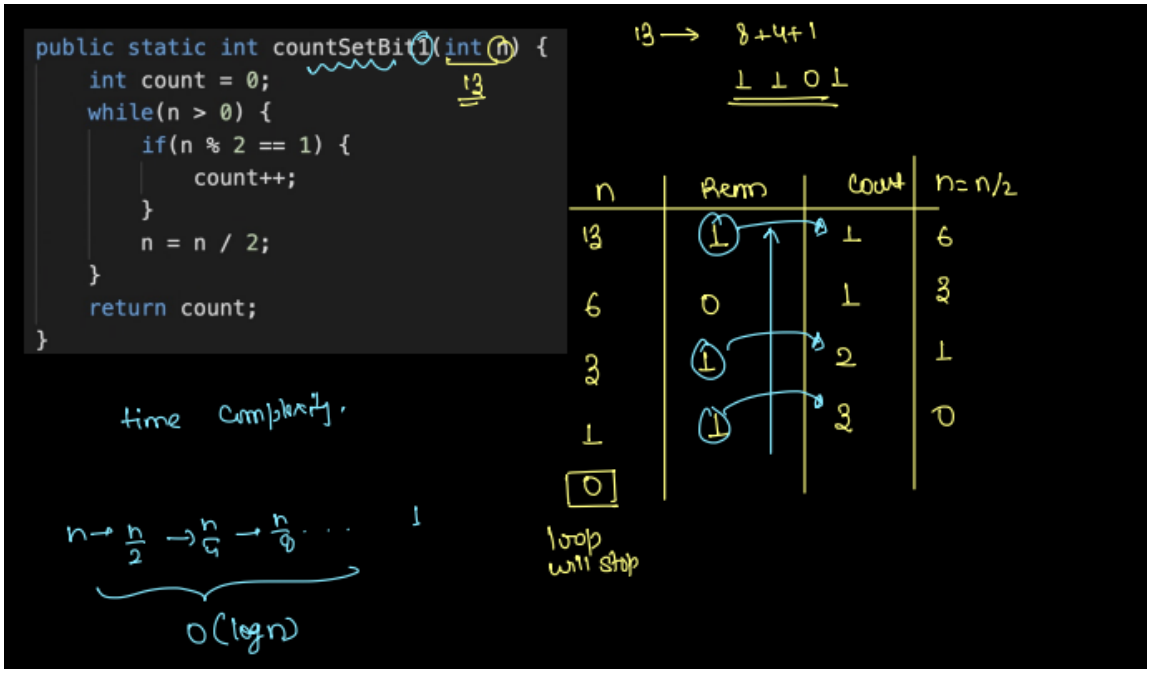


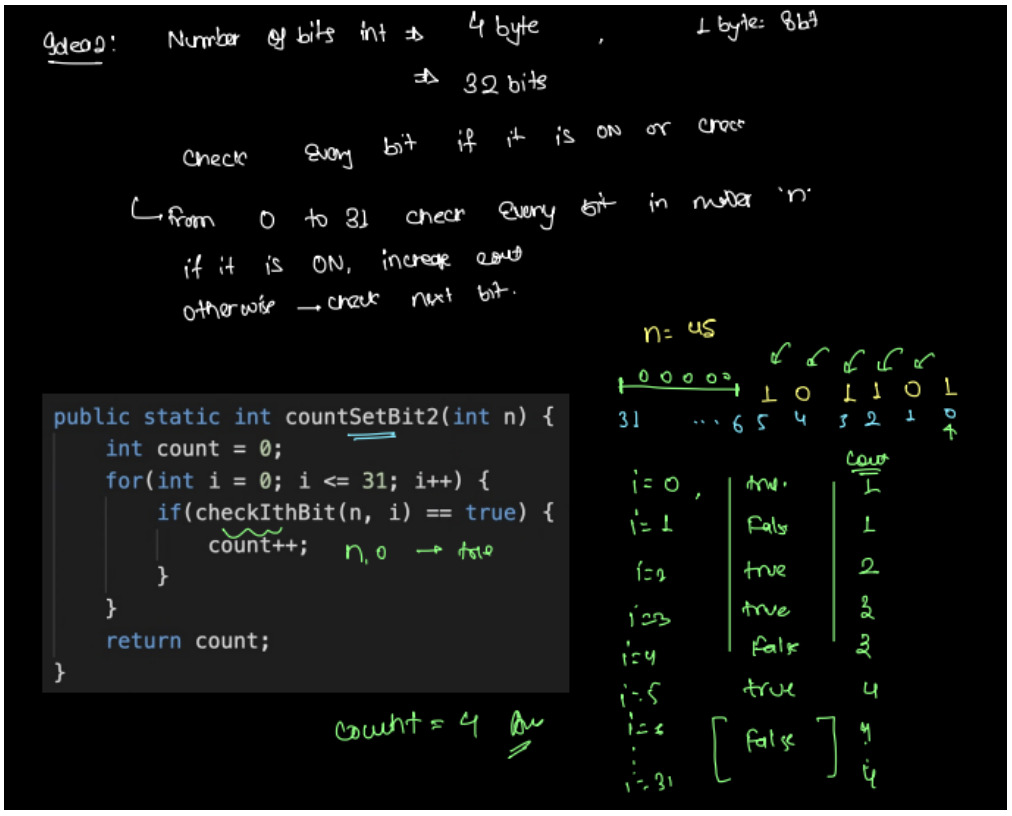

```java
public static boolean checkIthBit(int n, int i) {
    if((n & (1 << i)) != 0) {
        // if not 0, then bit is on
        return true;
    } else {
        // bit is OFF
        return false;
    }
}
```

Problem2: Given an integer number n, calculate total number of ON/SET bits [n is true]

n=45 -> 101101







Count Set bit 1

```java
public static int countSetBit1(int n) {
    int count = 0;
    while(n > 0) {
        if(n % 2 == 1) {
            count++;
        }
        n = n / 2;
    }
    return count;
}
```

T.C. : O(log n)

n = 8 = 2^3

gtr = $\log_2 2^3$

= 3 times.

n = 16 => 2^4

gtr = $\log_2 16 = \log_2 2^4$

= 4 times.

n = 2^{32}

gtr = $\log_2 2^{32}$ = 32 times.

Count Set bit 2

```java
public static int countSetBit2(int n) {
    int count = 0;
    for(int i = 0; i <= 31; i++) {
        if(checkIthBit(n, i) == true) {
            count++;
        }
    }
    return count;
}
```

T.C. : O(1)

32 times we one Iterating

n = 8 = 2^3

↳ gtr = 32 times.

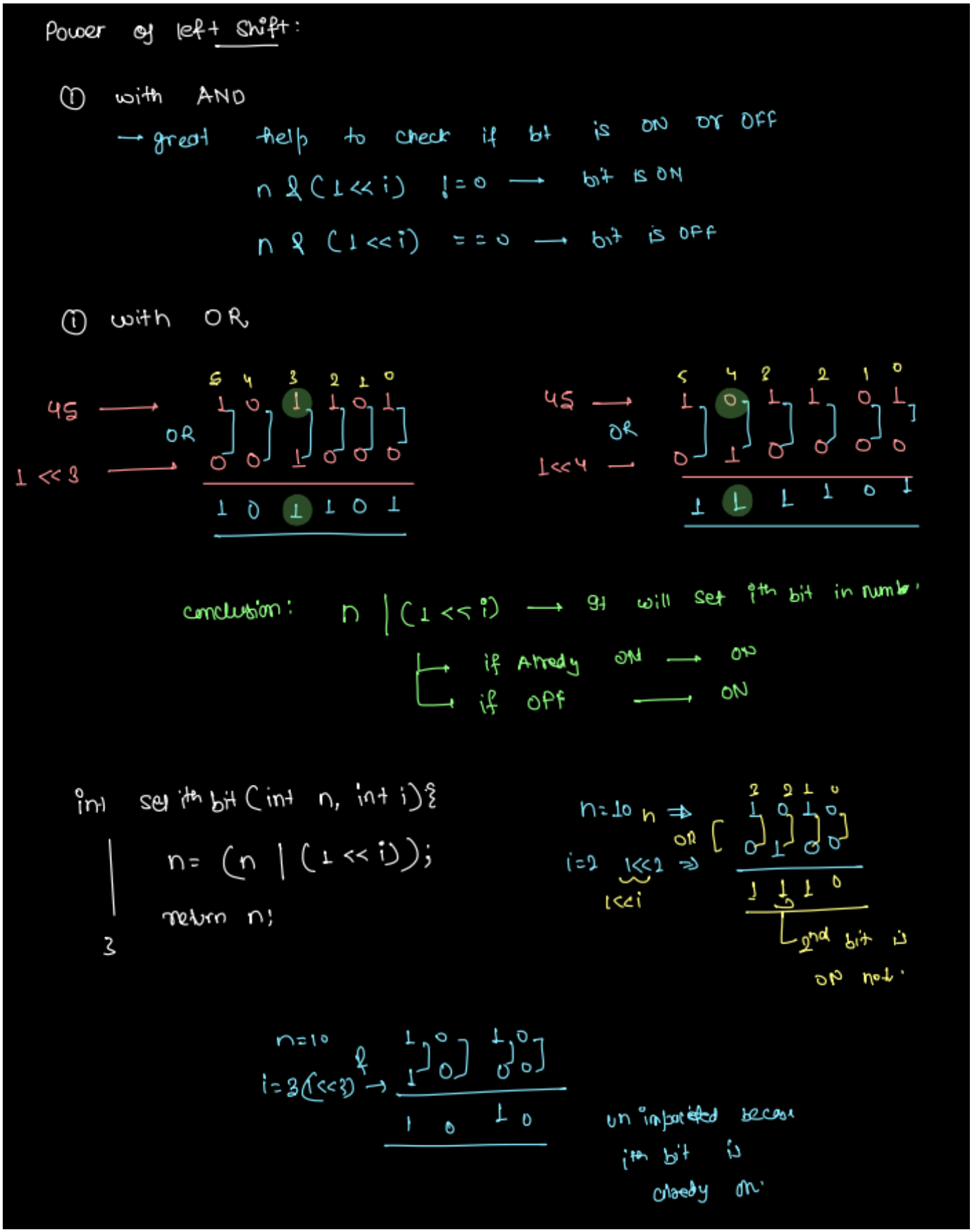
n = 16

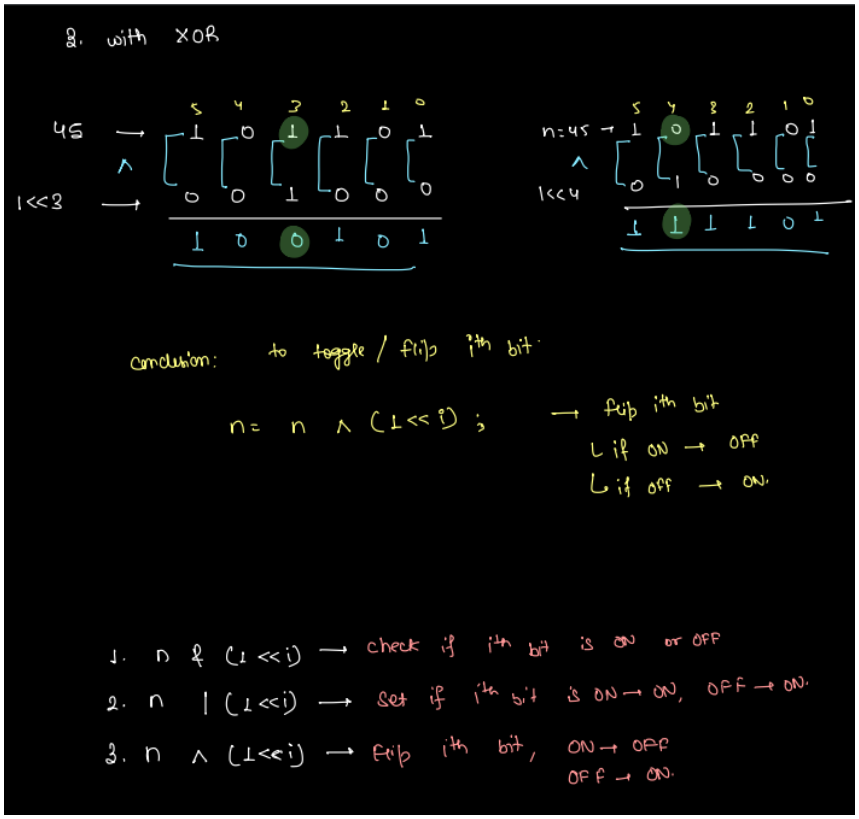
↳ gtr = 32 times

n = 2^{32}

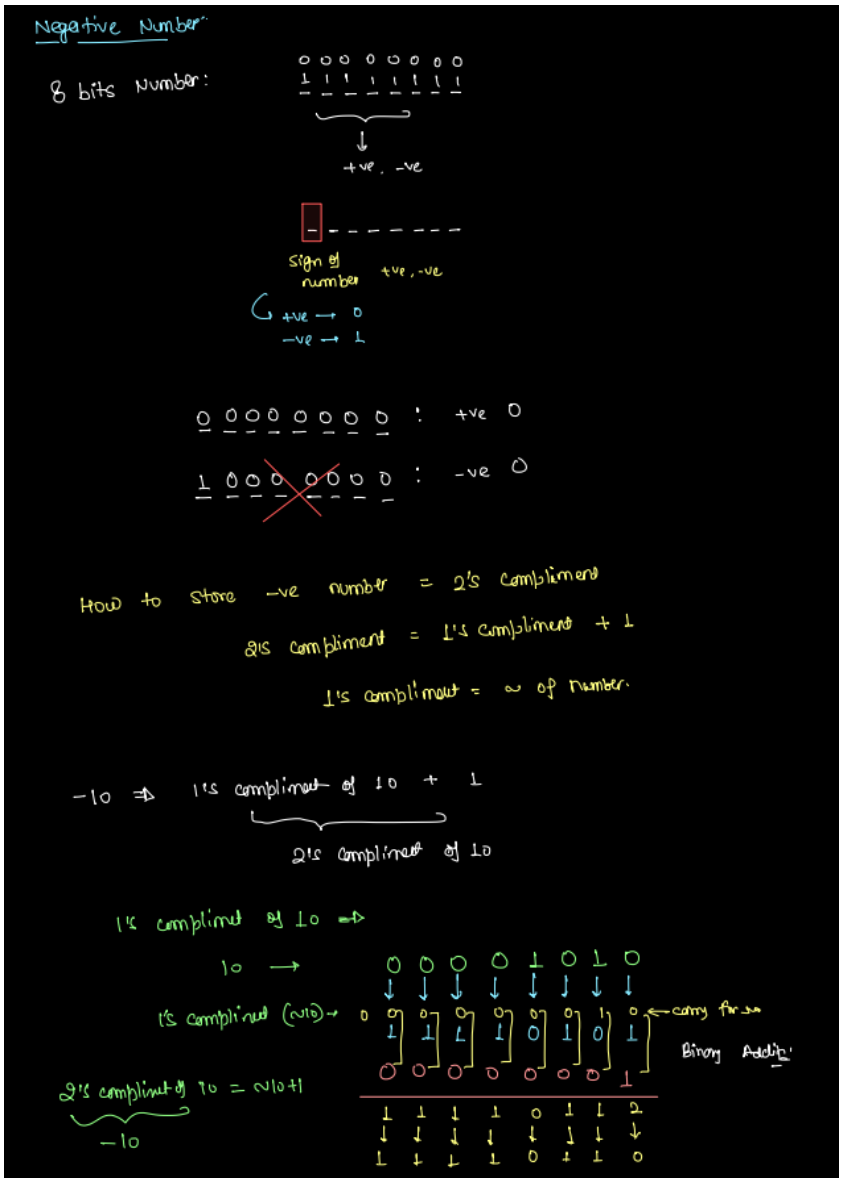
gtr = 32 times.

Power of left shift





Negative Number



$-10 \Rightarrow$ 1 1 1 1 0 1 1 0
(7 6 5 4 3 2 1 0)

Representation of sign bit.

$-1*2^7 + 1*2^6 + 1*2^5 + 1*2^4 + 0*2^3 + 1*2^2 + 1*2^1 + 0$

$= -128 + 64 + 32 + 16 + 0 + 4 + 2 + 0$

$= -128 + 118$

$= -10$ Ans

-45 = 1's compliment of 45 + 1

$45 \rightarrow$ 0 0 1 0 1 1 0 1

1's compliment of 45 = ~ 45

2's compliment of 45 = $-45 = \sim 45 + 1$

(Binary.)

$45 = 0\ 0\ 1\ 0\ 1\ 1\ 0\ 1$

$\sim 45 = 1\ 1\ 0\ 1\ 0\ 0\ 1\ 0$

$\sim 45 + 1 =$ 1 1 0 1 0 0 1 0
+ 0 0 0 0 0 0 0 1

$\sim 45 + 1 \Rightarrow$ 2's compl of 45 $= -45 =$ 1 1 0 1 0 0 1 1

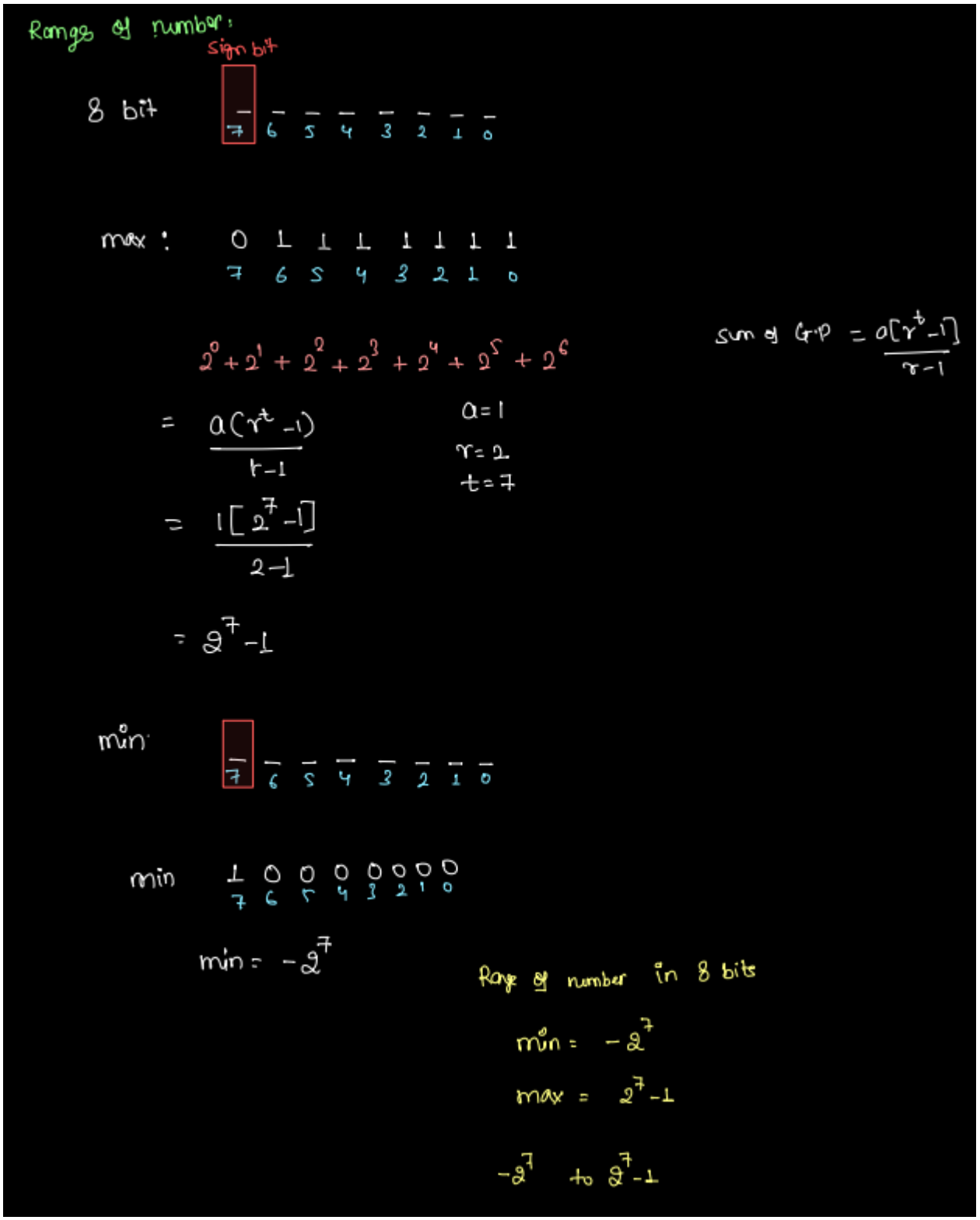
Sign bit: 1 1 0 1 0 0 1 1
(7 6 5 4 3 2 1 0)

$-1*2^7 + 1*2^6 + 0*2^5 + 1*2^4 + 0*2^3 + 0*2^2 + 1*2^1 + 1*2^0$

$= -128 + 64 + 0 + 16 + 0 + 0 + 2 + 1$

$= -128 + 83 = -45$ Ans

Range of number



32 bits

31	30	29	28	. . .	3	2	1	0
_	_	_	_	. . .	_	_	_	_

max :

31	30	29	28	. . .	3	2	1	0
0	1	1	1	. . .	1	1	1	1

$= \underbrace{2^0 + 2^1 + 2^2 + \dots + 2^{30}}$ (1, 1*2, 1*2²)

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

$a = 1$

$r = 2$

$t = 31$

$= \frac{1[2^{31} - 1]}{2 - 1}$

$= 2^{31} - 1$

$2^0 + 2^1 + 2^2 + \dots + 2^{30}$

$1 + 1*2^1 + 1*2^2 + \dots + 1*2^{30}$

$a + a*r + a*r^2 + \dots + a*r^{t-1}$

↳ $\frac{a(r^t - 1)}{t - 1} = \frac{1(2^{31} - 1)}{2 - 1}$

$= 2^{31} - 1$

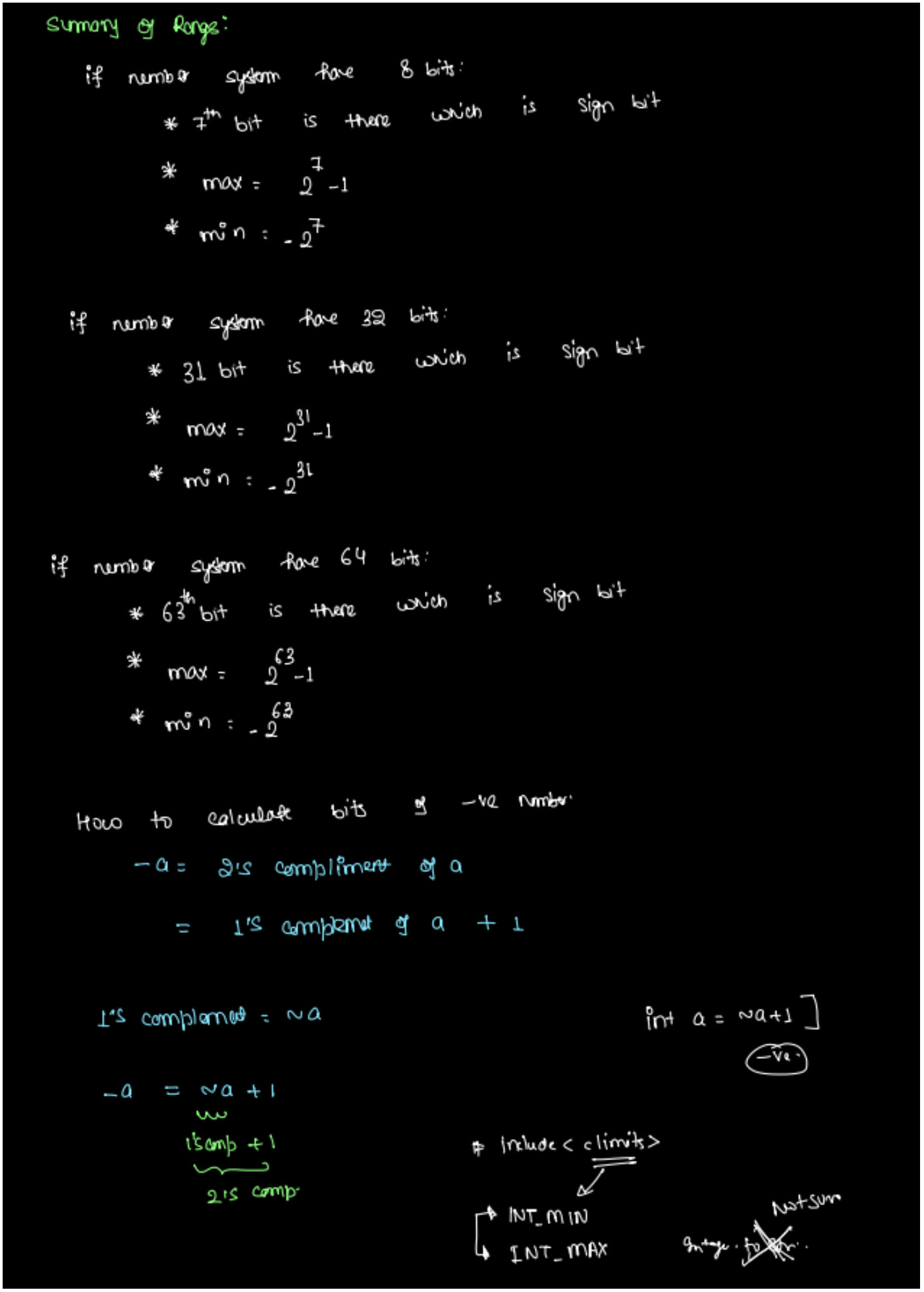
min :

31	30	29	28	. . .	3	2	1	0
1	0	0	0	. . .	0	0	0	0

$min = -2^{31}$

Range of integer number $= -2^{31}$ to $2^{31} - 1$

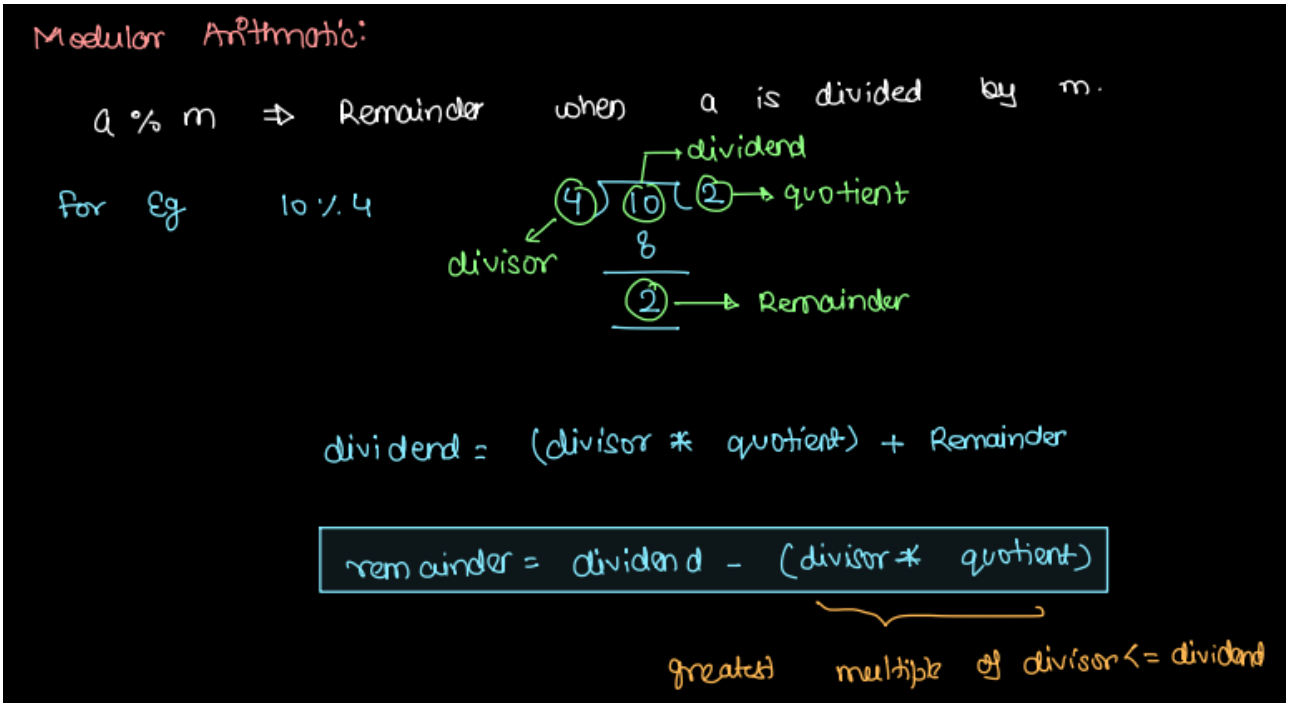
Summary of ranges

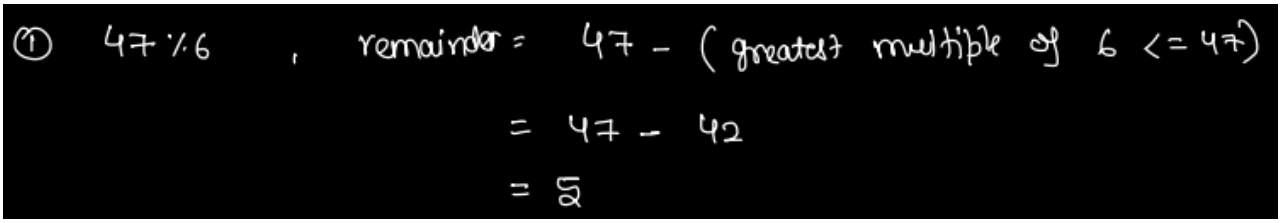


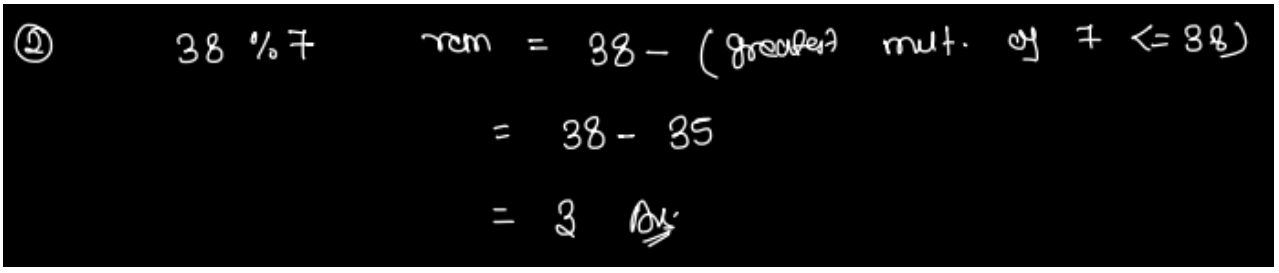
DSA: Subsets & Subsequences 16 - May 2023

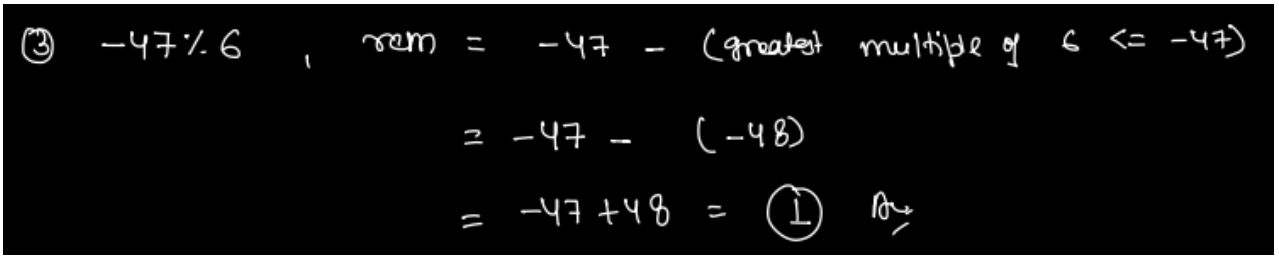
- Modular Arithmatic
- Properties of modulo
- calculate (a^n)%p
- Subsequence
- Subset
- Sum of Subset is K?
- Sum of Max from every subseq.

- **Modular Arithmatic**









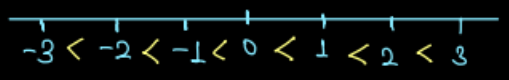
-

④ $-50 \% 6$, rem = -50 − (greatest mult. of 6 <= -50)

$= -50 - (-54)$

$= -50 + 54 = 4$ Ans

⑤ $-43 \% 7$, rem = -43 − (greatest mult. of 7 <= -43)



-7
-14
-21
-28
-35
-42
} Multiple of 7

-49
-56
-63
-70
. . .
} ⇒ -49 greatest number.

$= -43 - (-49)$

$= -43 + 49$

$= 6$ Ans

NOTE: Remainder is always +ve.

	google	python	Java
$-47 \% 6$	1	1	$-5 + 6 \Rightarrow 1$
$-50 \% 6$	4	4	$-2 + 6 \Rightarrow 4$
$-43 \% 7$	6	6	$-1 + 7 \Rightarrow 6$

```
if(rem <0){
    rem += divisor;
}
```

⊖47 % 6

= -47 % 6 = -5

-5 + 6 = (1)

Rem.

$-50 \% 6 \Rightarrow -50 \% 6 \Rightarrow -2 = 4$ Ans

$-2 + 6 = (4)$

```
if ( rem <0) {
    rem += divisor)
}
```

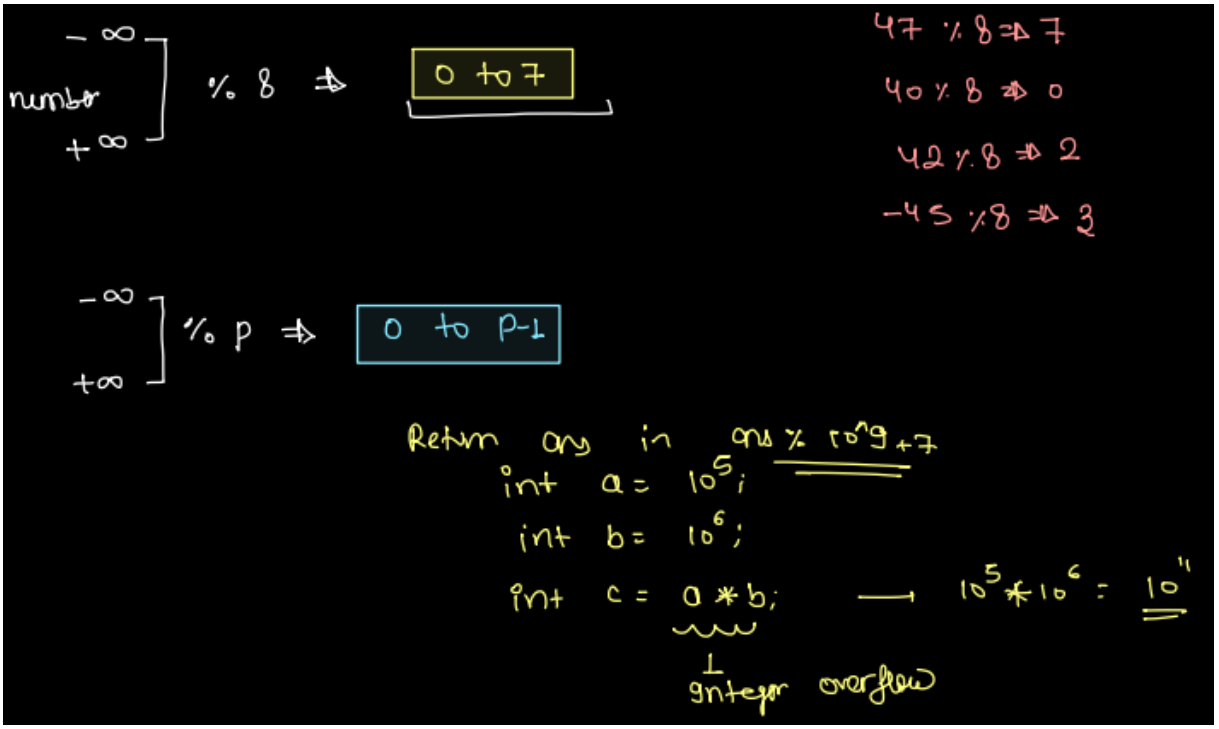
Return answer with ans % [$10^9 + 7$] → Range of ans ⇒ 0 to x-1

(x)

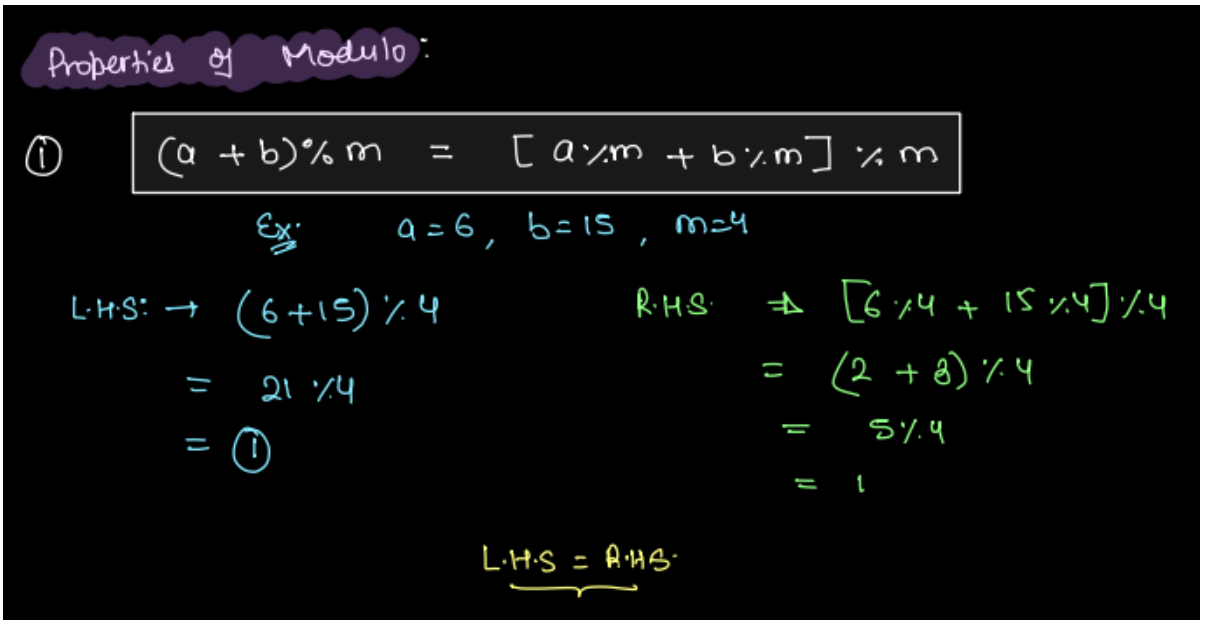
↓

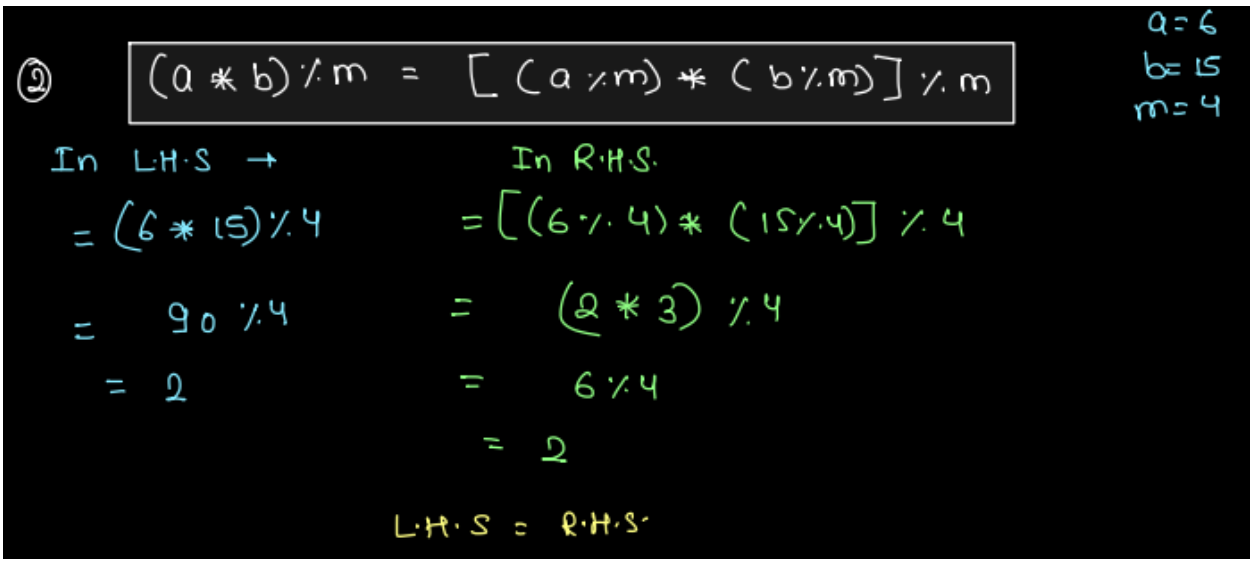
Why we do this?

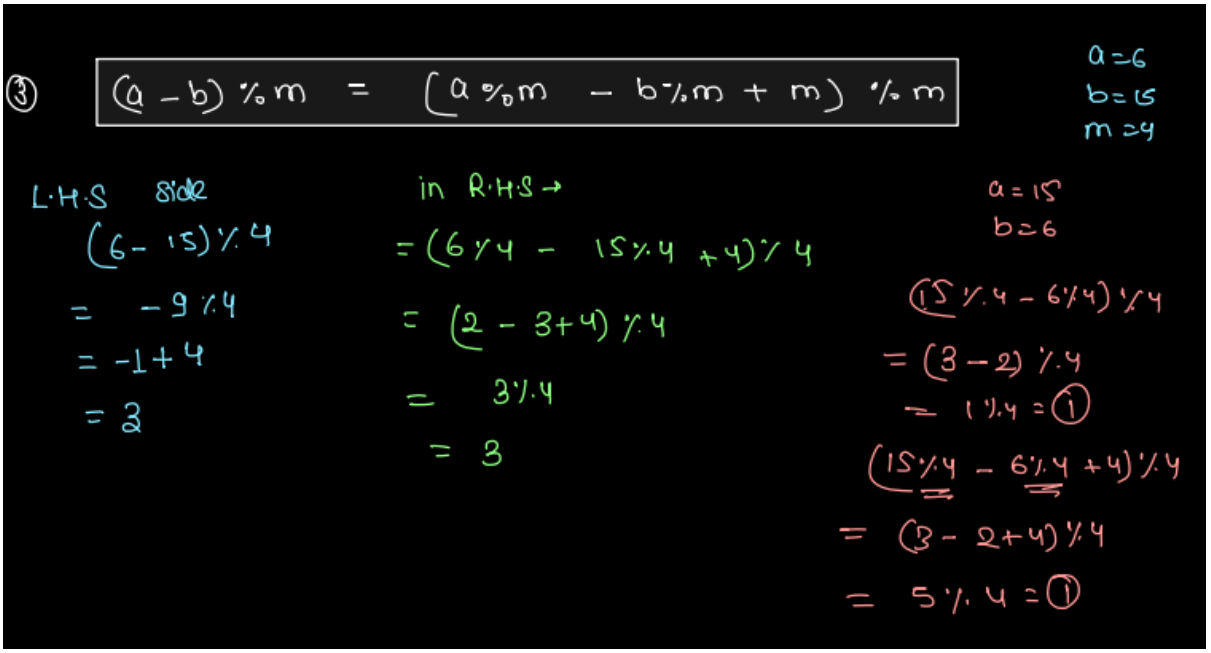
→ just to manage the overflow of answer.



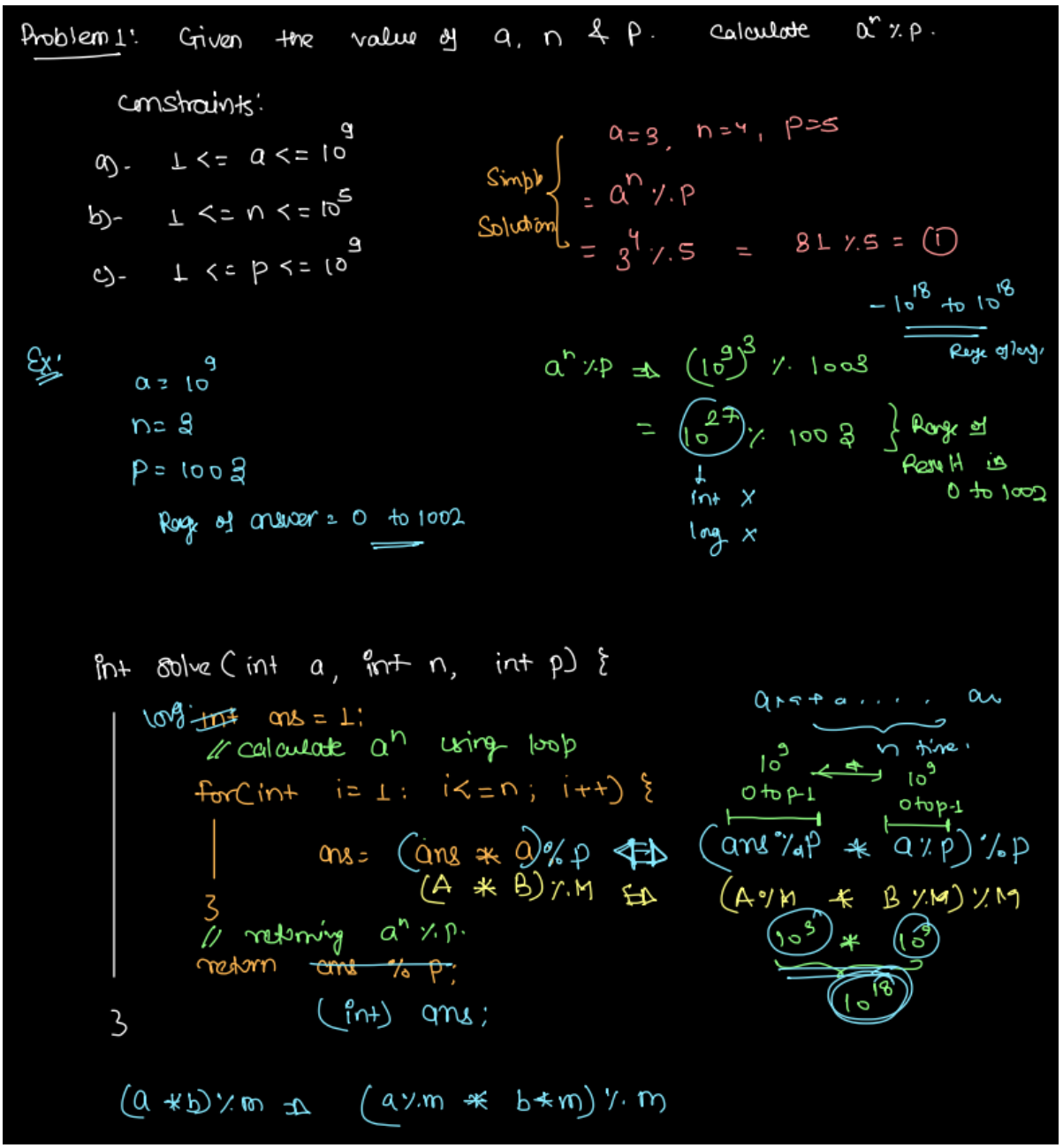
- **Properties of modulo**



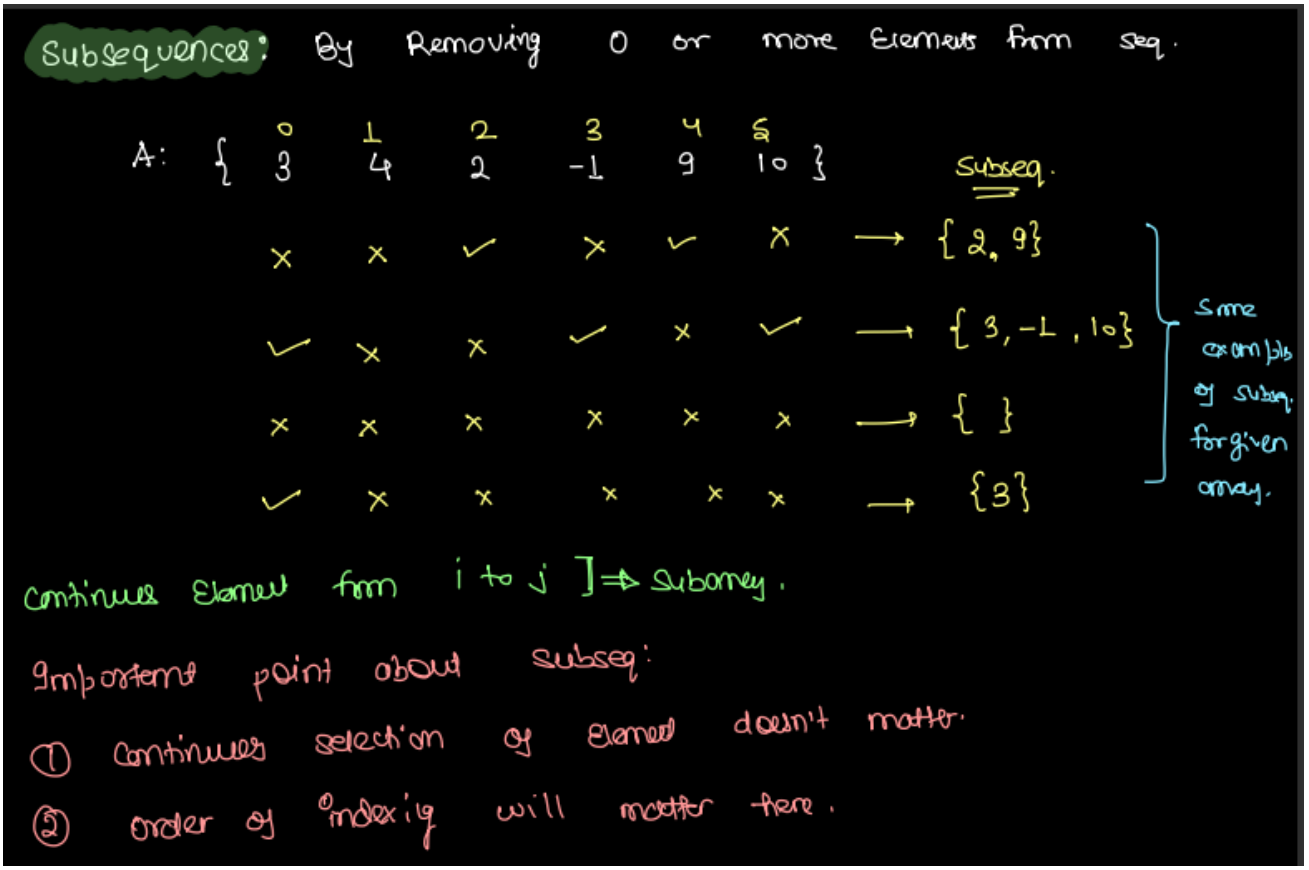


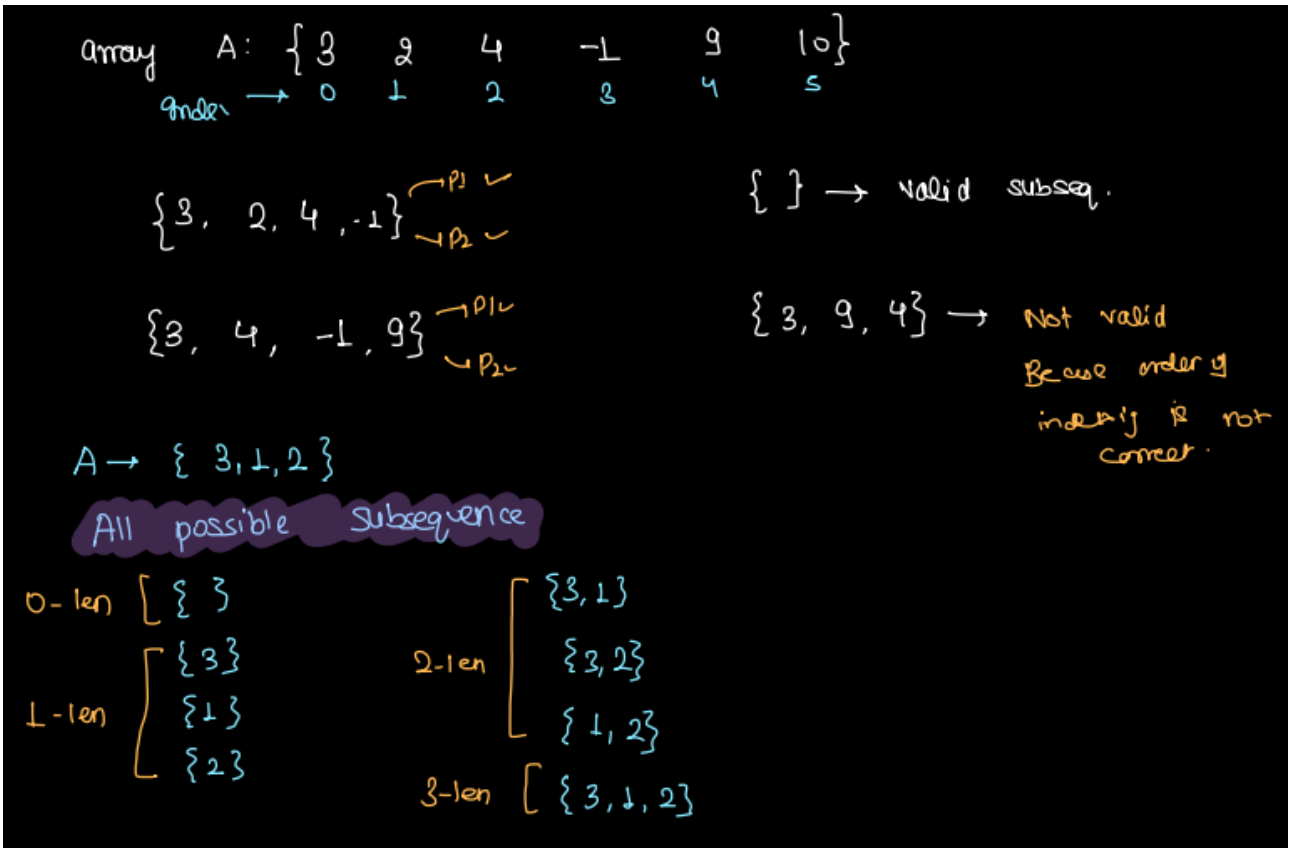


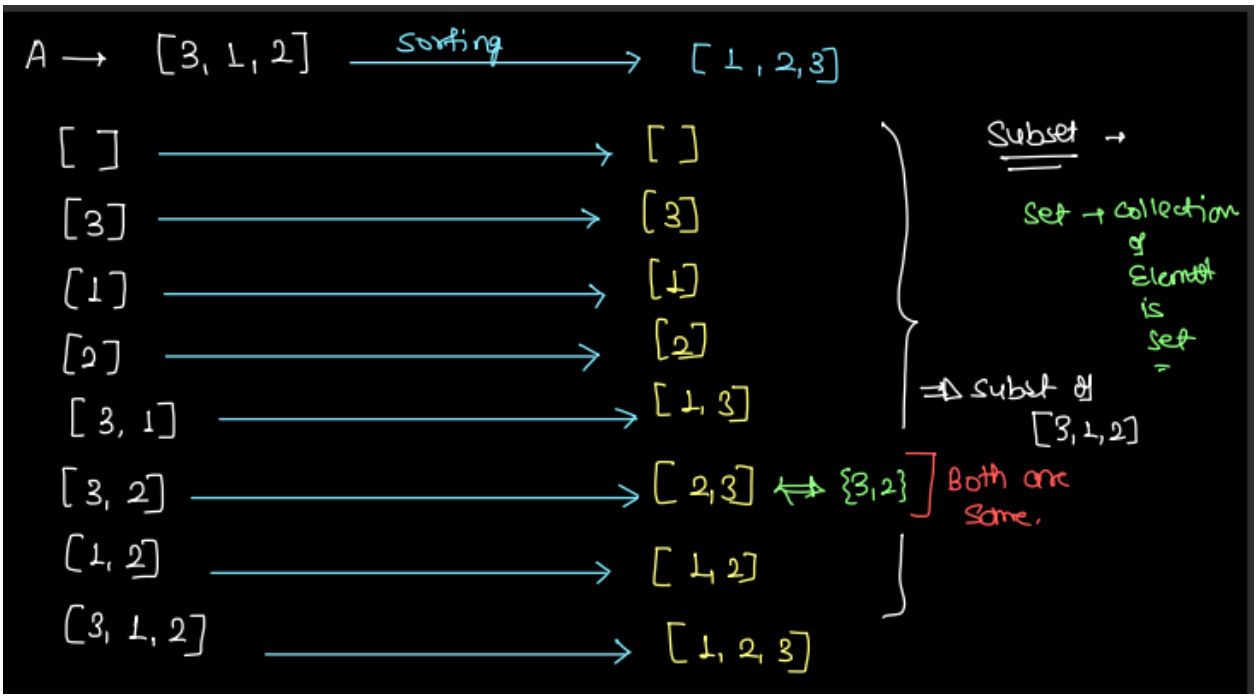
- **calculate (a^n)%p**



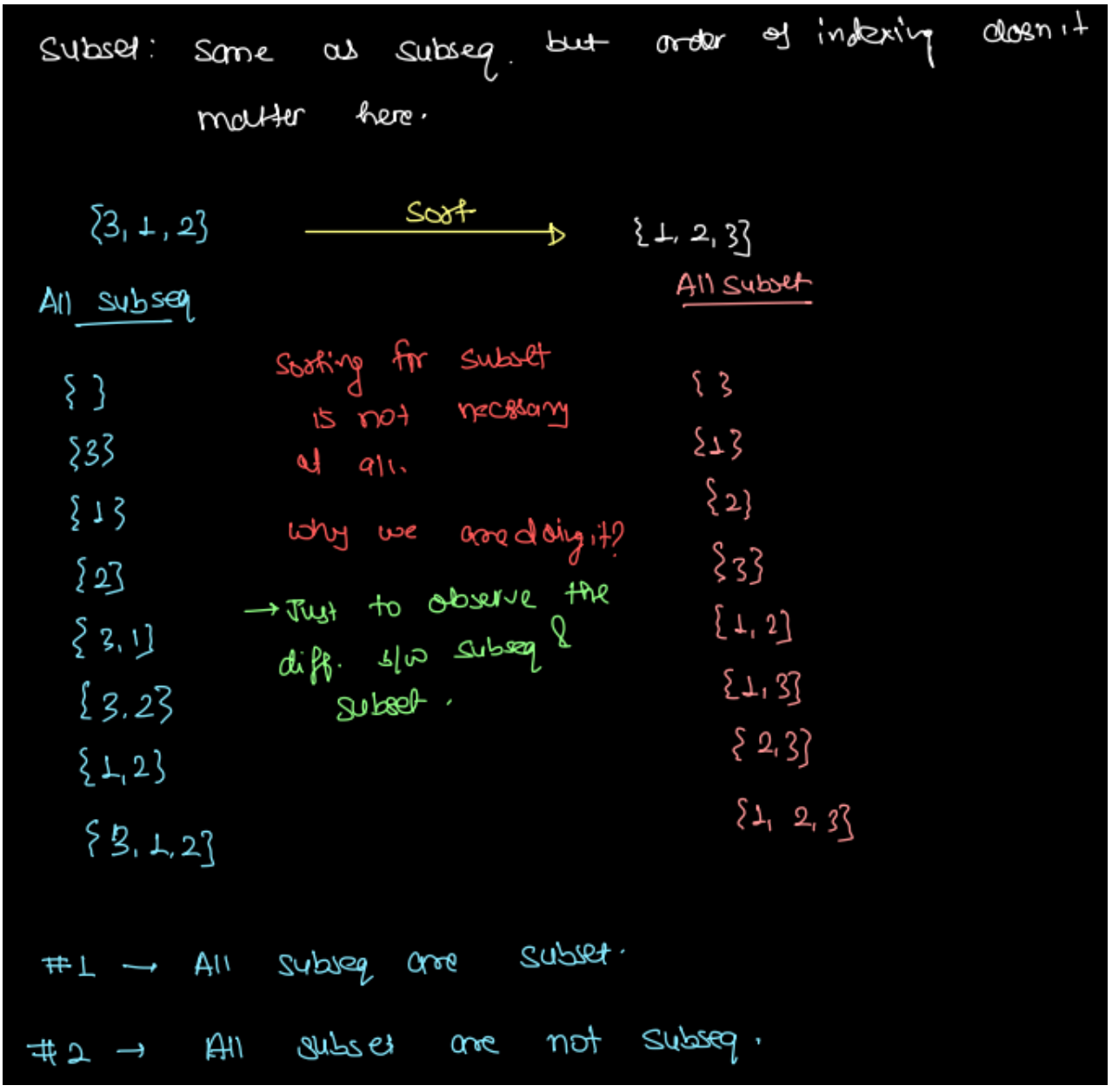
- **Subsequence**






A → [3, 1, 2]
Sorting
[1, 2, 3]
[ ] → [ ]
[3] → [3]
[1] → [1]
[2] → [2]
[3, 1] → [1, 3]
[3, 2] → [2, 3] ⟺ {3, 2}
[1, 2] → [1, 2]
[3, 1, 2] → [1, 2, 3]
Subset →
Set → collection of Element is set
⇒ subset of [3, 1, 2]
Both are same.

- **Subset**


Subset: Same as subseq. but order of indexing doesn't matter here.
{3, 1, 2}
Sort
{1, 2, 3}
All subseq
{ }
{3}
{1}
{2}
{3, 1}
{3, 2}
{1, 2}
{3, 1, 2}
Sorting for subset is not necessary at all.
Why we are doing it?
→ Just to observe the diff. b/w subseq & subset.
All Subset
{3}
{1}
{2}
{3}
{1, 2}
{1, 3}
{2, 3}
{1, 2, 3}
#1 → All subseq are subset.
#2 → All subset are not subseq.

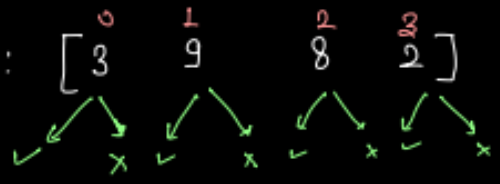
A: [3 2 4 -1 9 10]
0 1 2 3 4 5

Eg → ① [3 2 4 9] ⟶ valid subseq + valid subset

② [3 9 4] ⟶ Invalid subseq + valid subset

count of subsequence:

A: [3 9 8 2]



$2 \times 2 \times 2 \times 2 = 2^4 = 16$

If length of array is n.

total number of subseq = 2^n

total number of subset = 2^n

All the elements are distinct in subset

NOTE:
Whenever we deal with subset → inputs are distinct elements.

$\{1_1, 1_2\}$

Subseq. →

All Subseq.
- $\{\}$
- $\{1_1\}$
- $\{1_2\}$
- $\{1_1, 1_2\}$

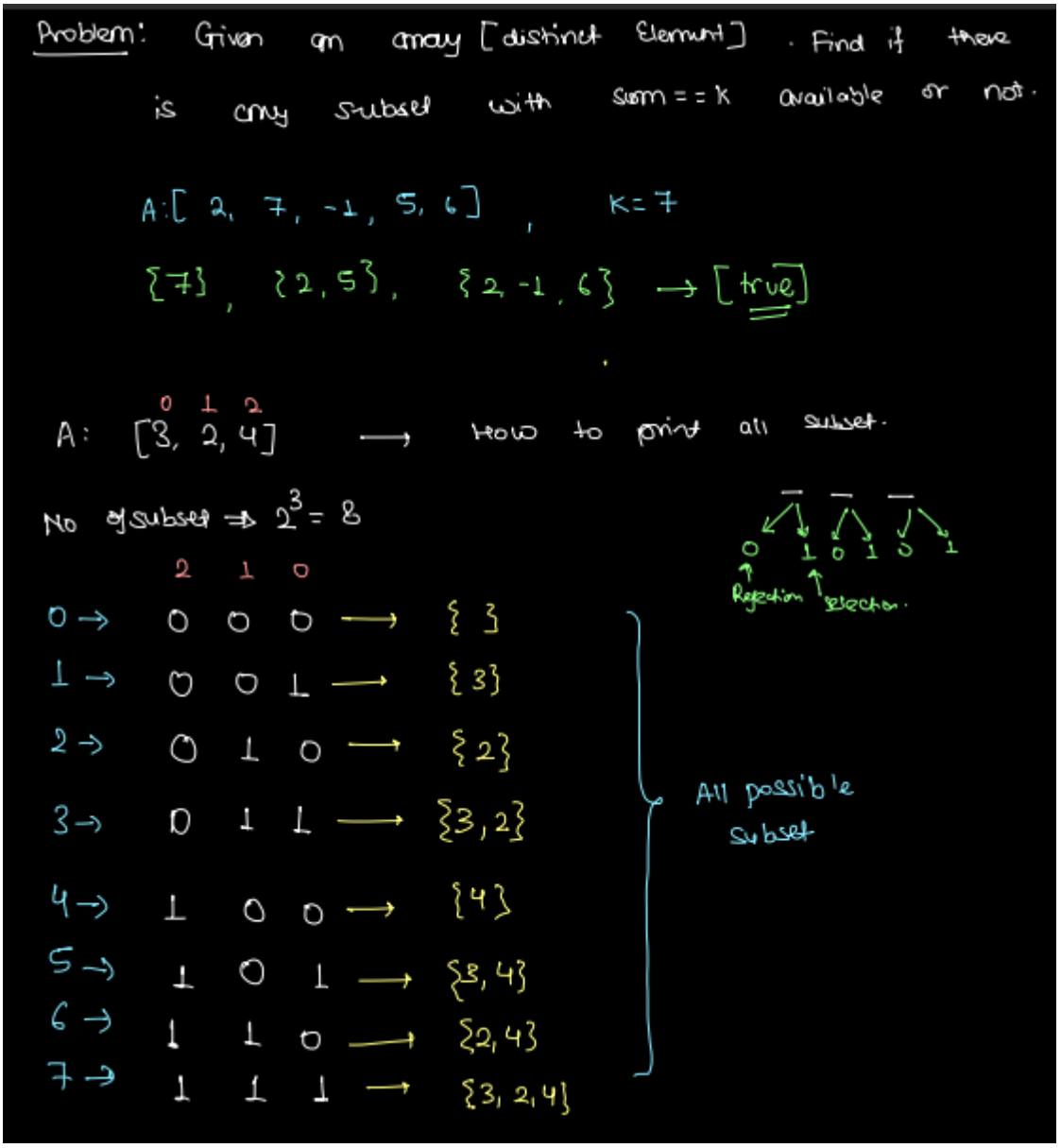
All subset
- $\{\}$
- $\{1_1\}$
- ~~$\{1_2\}$~~
- $\{1_1, 1_2\}$

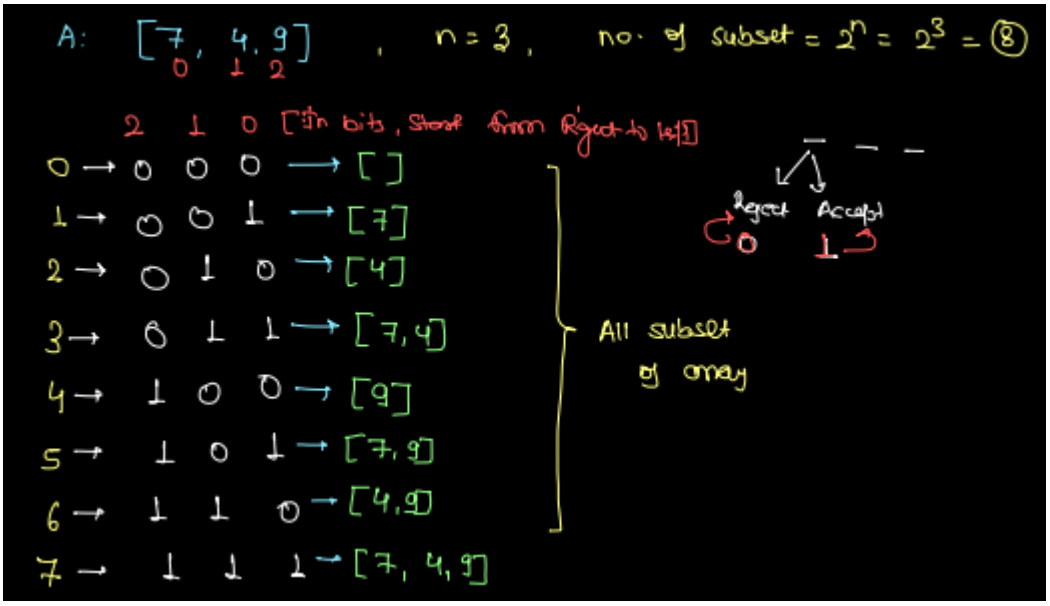
} 3 subset

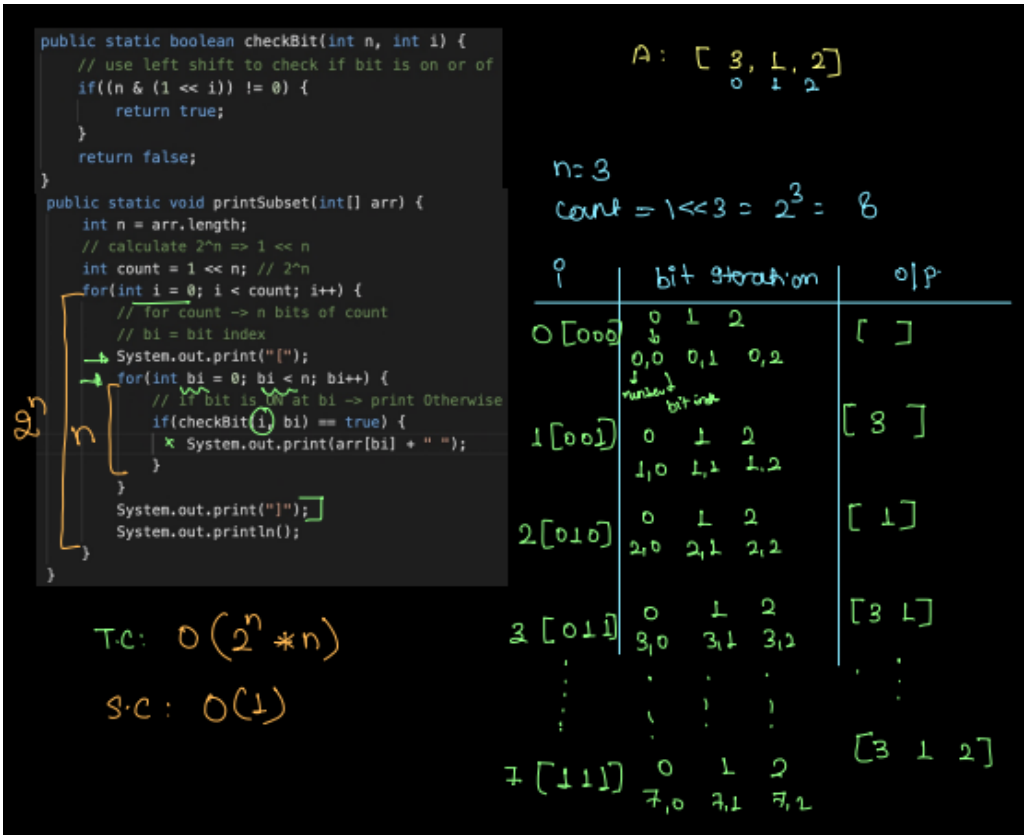
Permutation & Combination

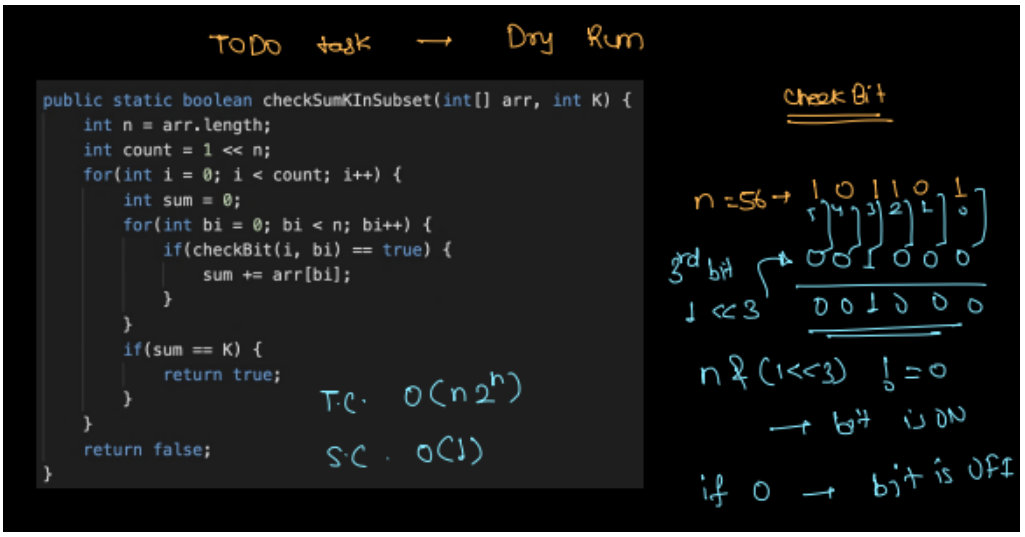
→ Mathematics

- **Sum of Subset is K?**

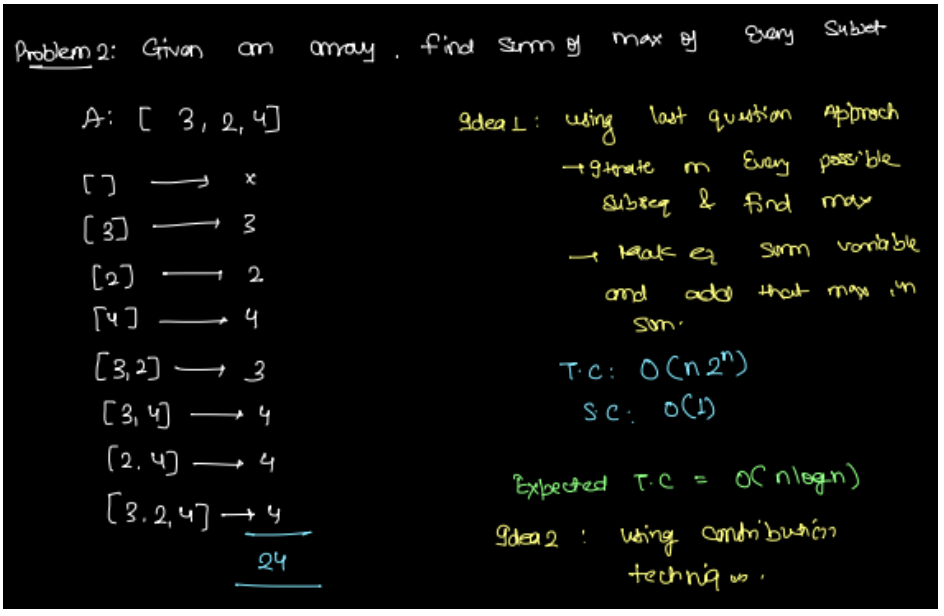


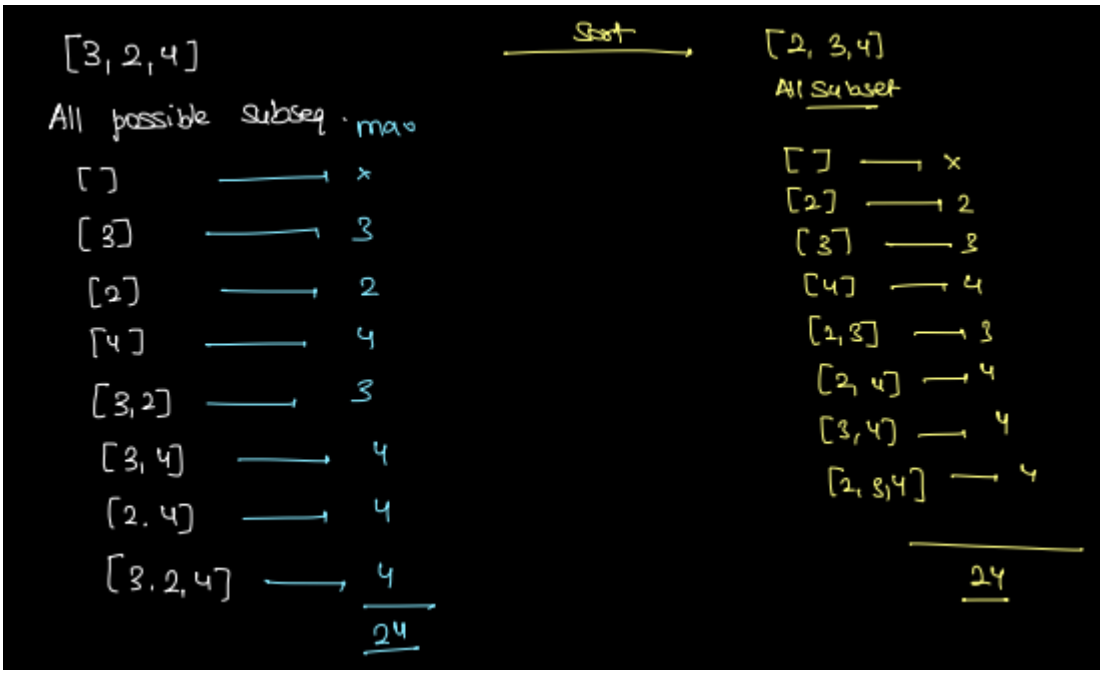


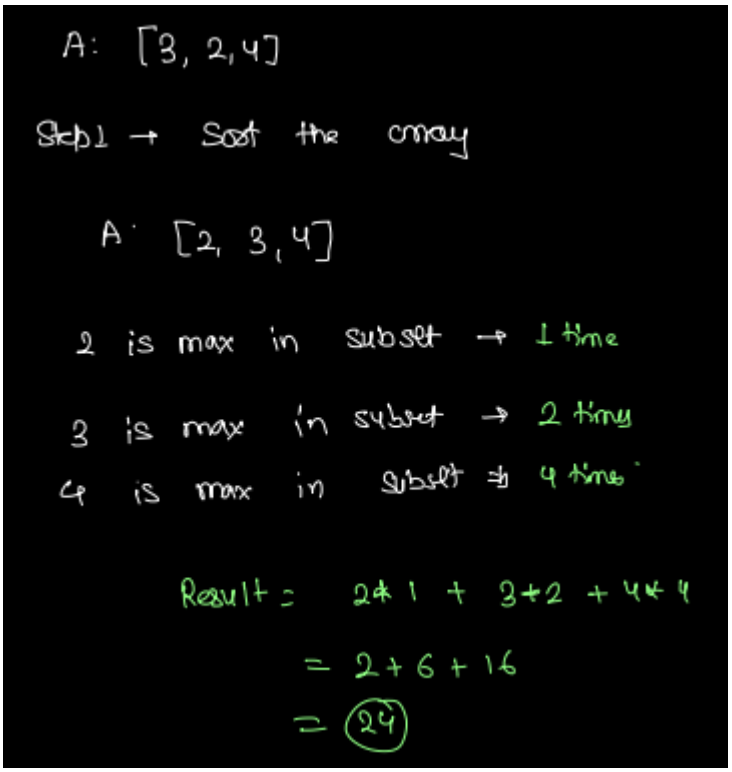


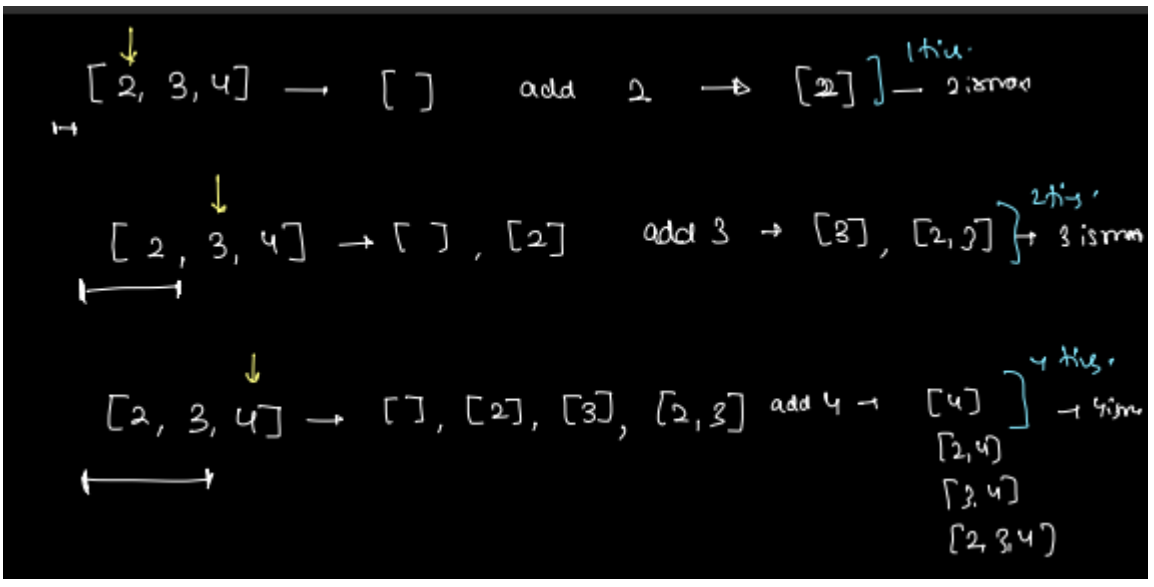


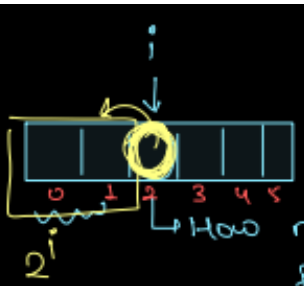
- **Sum of Max from every subseq.**




[3, 2, 4]
Sort
[2, 3, 4]
All possible subseq · max
[] → x
[3] → 3
[2] → 2
[4] → 4
[3, 2] → 3
[3, 4] → 4
[2, 4] → 4
[3, 2, 4] → 4
24
All Subset
[] → x
[2] → 2
[3] → 3
[4] → 4
[2, 3] → 3
[2, 4] → 4
[3, 4] → 4
[2, 3, 4] → 4
24


A: [3, 2, 4]
Step 1 → Sort the array
A: [2, 3, 4]
2 is max in subset → 1 time
3 is max in subset → 2 time
4 is max in subset → 4 time
Result = 2*1 + 3*2 + 4*4
= 2 + 6 + 16
= 24


[2, 3, 4] → [] add 2 → [2] 1 time. 2 is max
[2, 3, 4] → [], [2] add 3 → [3], [2, 3] 2 time. 3 is max
[2, 3, 4] → [], [2], [3], [2, 3] add 4 → [4], [2, 4], [3, 4], [2, 3, 4] 4 time. 4 is max



How many subset is possible from 0 to i-1

Element count in left = i

Number of subset from 0 to i-1 $\Rightarrow$ 2^i

[1, 2, 3, 7]

i = 0	i = 1	i = 2	i = 3
1 is max	2 is max	3 is max	7 is max
Subset in left side			
2^0 = 1	2^1 = 2	2^2 = 4	2^3 = 8
1 * 1	+ 2 * 2	+ 3 * 4	+ 7 * 8

$\sum_{i=0}^{n-1} 2^i * arr[i]$

1 + 4 + 12 + 56 = 73

```
int Sum (int[] arr) {
    Array.sort (arr);                          // nlogn
    int Sum=0;
    for(int i=0; i< arr.length; i++) {         // n
        Sum += arr[i] * (1<<i);                // 2^i
    }
    return sum;
}
```

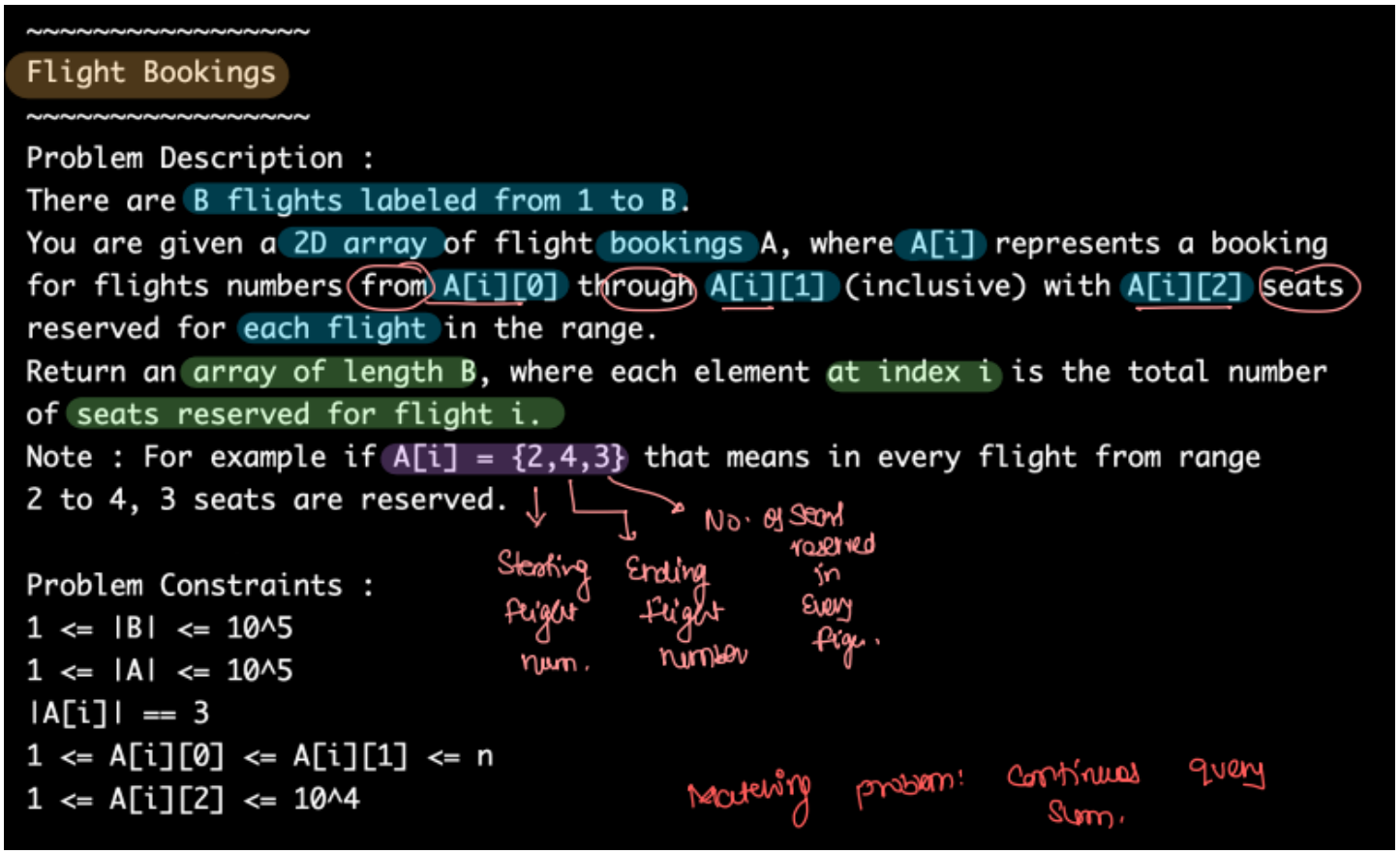
TODO. Manage modulus Arithmatic here.

total T.C = nlogn + n

Big-O = O(nlog)

S.C = O(1)

DSA: Contest 2 Discussion - 18 - May 2023



https://www.interviewbit.com/snippet/8eaa6bf1aad5b4adc0e7/

Main Logic:

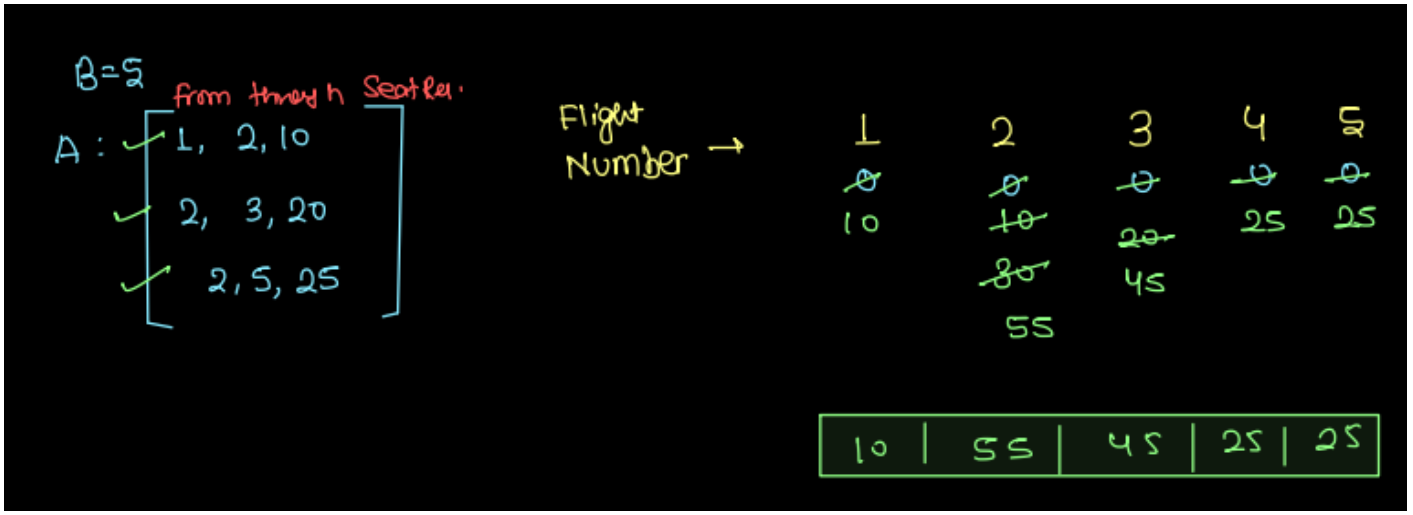
Brute force

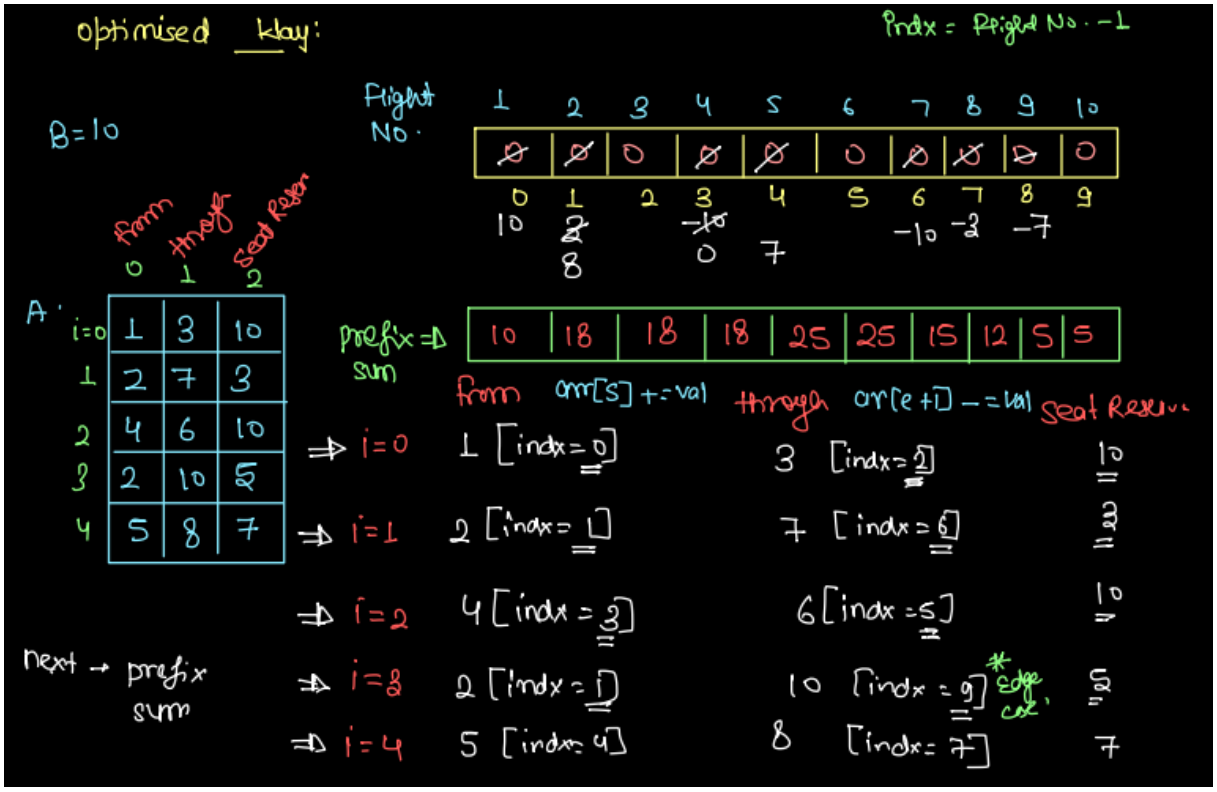
1. Create an array called "ans" with a size of B.
2. Iterate through numbers from 0 to A.length - 1.
3. For each iteration, define three integer variables: "from," "to," and "seats" and assign values to them as follows:
    a. "from" gets the value of A[i][0] - 1 (subtracting 1 because the query array A uses 1-based indexing).
    b. "to" gets the value of A[i][1] - 1 (again, subtracting 1 for 0-based indexing).
    c. "seats" gets the value of A[i][2].
4. Run another loop ranging from "from" to "to" (inclusive) and add the value of "seats" to the "ans" array for each index.
5. Finally, return the "ans" array after both loops have completed.

Optimized Approach: In this optimized approach, we'll use a simpler logic. We'll add to the first index and subtract from the lastIndex + 1 since the array already uses 1-based indexing.

Here are the steps:

1. Create an array called "ans" with a size of B.
2. Iterate through numbers from 0 to A.length - 1.
3. For each iteration, define three integer variables: "from," "to," and "seats," and assign values to them as follows:
    a. "from" gets the value of A[i][0] - 1 (subtracting 1 because the query array A uses 1-based indexing).
    b. "to" gets the value of A[i][1] (no need to subtract 1 here since it's already 1-based indexing).
    c. "seats" gets the value of A[i][2].
4. Update the "ans" array as follows:
    a. Add "seats" to "ans[from]" (A[start] += seats).
    b. If "to" is less than B, subtract "seats" from "ans[to]" (if to < B then A[to] -= seats).
5. Run another loop from 1 to B - 1 (outside of the previous loop) and update the "ans" array to store cumulative sums using: "ans[i] = ans[i-1] + ans[i]".
6. Finally, return the "ans" array after both loops have completed.





Steps:

① Create an array of length B having initial elements are 0.

② Extract query i:

Start = A[i][0], si = start - 1

end = A[i][1], ei = end - 1

seats = A[i][2]

③ For every query, ans[si] += seats.

if (ei + 1 < n) {

ans[ei + 1] -= seats.

④ In end, calculate prefix sum & return ans array.

```java
public int[] solve(int[][] A, int B) {
    // 1. create an array for answer
    int[] ans = new int[B];
    // 2. According to flight booking deatils,
    // update answer array for every query
    for(int i = 0; i < A.length; i++) {
        int si = A[i][0] - 1;
        int ei = A[i][1] - 1;
        int seats = A[i][2];

        ans[si] += seats;
        if(ei + 1 < B) {
            ans[ei + 1] -= seats;
        }
    }
    // 3. calculate prefix Sum
    for(int i = 1; i < B; i++) {
        ans[i ] = ans[i] + ans[i - 1];
    }
    // 4. return answer array
    return ans;
}
```

Number of query => q

T.C = O(q)

Overall T.C O(q+n)

Number of flight → n

T.C = O(n)

Construct Binary Number

Problem Description :

Construct a binary number having A 1's followed by B 0's. Return the decimal value of that binary number.

For eg :

A = 3, B = 2

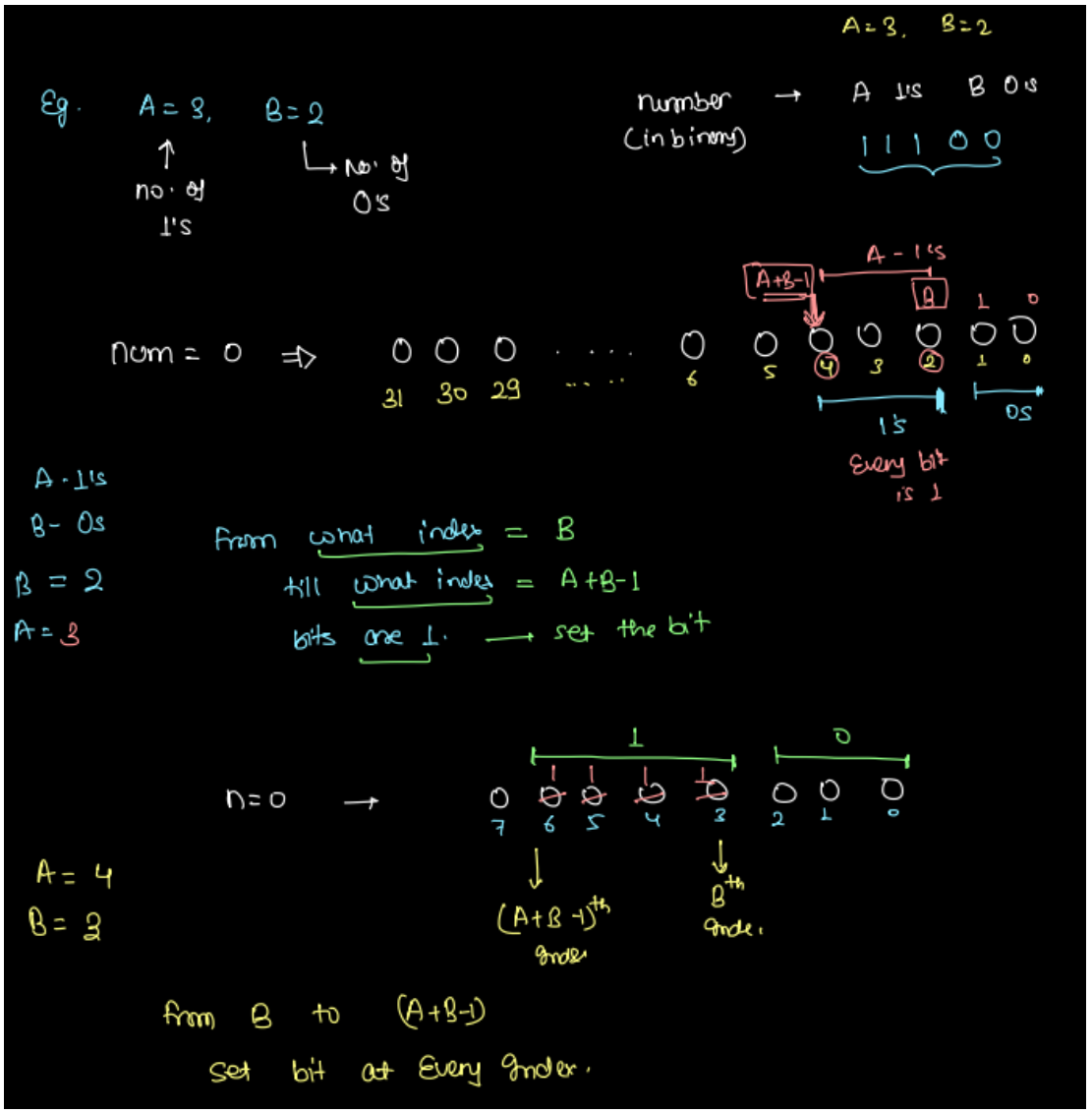
Answer = (11100)2 . Return = 28

Problem Constraints :

1 <= A + B <= 30

Main Logic:

1. Initialize a variable called "num" and set its value to 0.
2. Execute a loop from 0 to less than the sum of A and B.
    a. Inside the loop, update the "num" variable using the bitwise OR operation: "num |= 1 << i".
3. Return the "num" variable as the result.



How to set ith bit in number

n = 1 0 0 1 0 1 (bits 5 4 3 2 1 0)

Set 3rd index bit

1<<3 = 0 0 1 0 0 0

n = n | (1<<3) = 1 0 1 1 0 1

↳ 3rd bit is ON

How to set ith bit in n.

$$n = [n \mid (1 << i)]$$

```
int n = 0;
for(int i = B; i <= (A+B-1); i++){
    n = (n | (1<<i));
}
return n;
```

A=3 B=2

0 0 0 0 0 ←

A+B-1 = 3, B = 2

from what index till what index you have to set bit as 1

A = 4 B = 2

from B = 2 to A+B-1 = 5

```
public int solve(int A, int B) {
    int n = 0;
    for(int i = B; i <= (A + B - 1); i++) {
        n = (n | (1 << i));
    }
    return n; → ans.
}
```

n = 0

	5	4	3	2	1	0	
n =	0	0	0	0	0	0	= 0
1<<2	0	0	0	1	0	0	
n =	0	0	0	1	0	0	= 4
1<<3	0	0	1	0	0	0	
n =	0	0	1	1	0	0	= 12
1<<4	0	1	0	0	0	0	
n =	0	1	1	1	0	0	= 28
1<<5	1	0	0	0	0	0	

i=2 (1<<2)

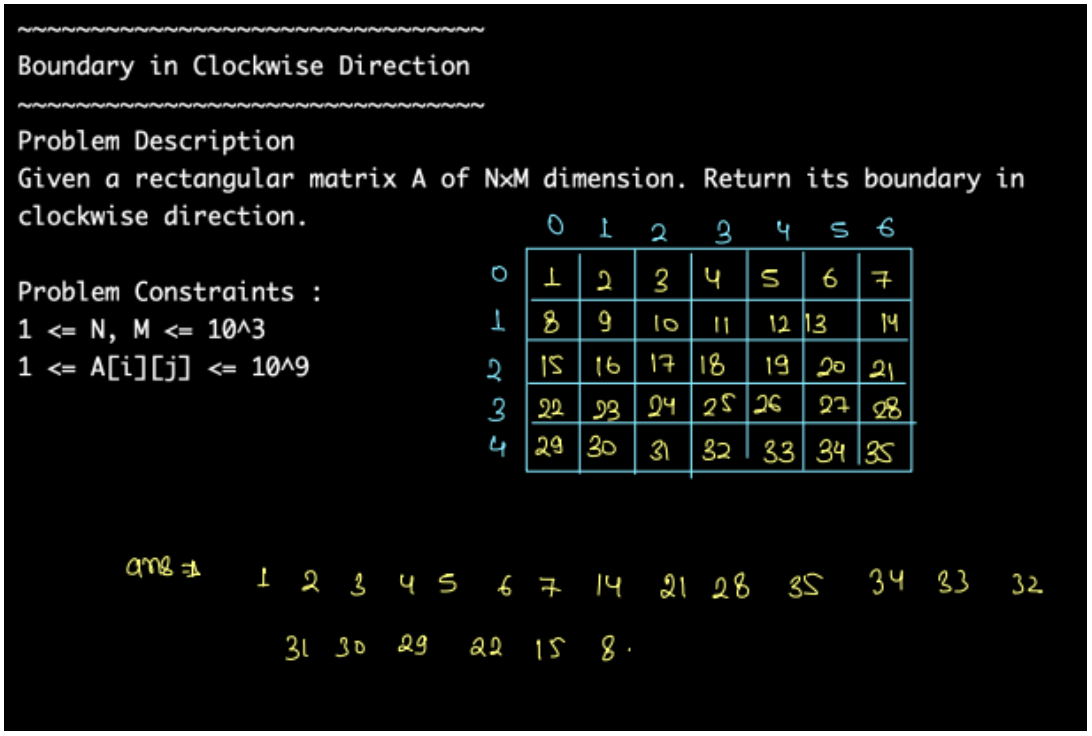
i=3 (1<<3)

i=4 (1<<4)

i=5 (1<<5)

Final ans ⇒ n = 1 1 1 0 0

originally n is decimal number sum



https://www.interviewbit.com/snippet/bcd7f1bd12700b1fb7ba/

Main Logic:

1. Initialize four variables: "n," "m," "i," and "j," and set their initial values to the length of the matrix, the length of the first row of the matrix, 0, and 0, respectively.
2. Create an array called "ans" with a length of 2 * (n + m - 2), and initialize an index variable "idx" to 0.
3. Run the first for loop to print the top row:
    a. Initialize a variable "k" to "j".
    b. Iterate while "k" is less than "m - 1" (subtracting 1 because we print one item less in this row).
    c. Inside the loop, assign "mat[i][j]" to "ans[idx]" and increment "idx."
    d. Increment "j."
4. Run the second for loop to print the right column boundary:
    a. Initialize a variable "k" to "i."
    b. Iterate while "k" is less than "n - 1."
    c. Inside the loop, assign "mat[i][j]" to "ans[idx]" and increment "idx."
    d. Increment "i."
5. Run the third for loop to print the bottom row boundary:
    a. Initialize a variable "k" to "j."
    b. Iterate while "k" is greater than 0.
    c. Inside the loop, assign "mat[i][j]" to "ans[idx]" and increment "idx."
    d. Decrement "j."
6. Run the fourth for loop to print the left column boundary:

a. Initialize a variable "k" to "i."
b. Iterate while "k" is greater than 0.
c. Inside the loop, assign "mat[i][j]" to "ans[idx]" and increment "idx."
d. Decrement "i."

7. return ans array

This logic traverses the boundaries of the matrix and populates the "ans" array with the corresponding values.