

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  
$$b - a + 1 = \text{No. of iterations}$$
- Exclusive bracket  
$$b - a = \text{No. of iterations}$$
- Common Ratio  
$$a * r / a = r (\text{ratio})$$
 here **a** is first number and **r** is second number in list  
$$r / a = \text{common ration}$$
 - get common ratio
- Sum of N(natural numbers) term in the GP (Geometric progression)  
$$Sn = a(r^n - 1) / r - 1; a \text{ is first term (number)} r \text{ common ration}$$
- 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  
$$\text{prefixSum}[i] = \text{prefixSum}[i-1] + A[i]$$
- Keep in mind ceil, floor, pow will always return double
- if power is minus then we will write like below.  
$$2^{-1} = \frac{1}{2^1} \rightarrow \frac{1}{2} = .5$$

it means we will add batta of 1 and power in num below bata

$$a^{-b} = \frac{1}{a^b} \quad | \quad 2^{-3} = \frac{1}{2^3} = \frac{1}{8} = 0.125$$

### 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  $1 + 2 + 3 + \dots + n$ , we have a special formula to find the answer:  $n(n+1)/2$ .
- Similarly, if we're adding numbers from  $1 + 2 + 3 \text{ up to } (n-1)$ , 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

Problem 1: Given a number  $N$ , count factor of  $N$ .

$24 \rightarrow 1, 2, 3, 4, 6, 8, 12, 24$ , count = 8

Observation :  
① Smallest factor = 1  
② Largest factor =  $N$   
③ Factor: if  $N \% i == 0$ , ' $i$ ' is factor of  $N$ .  
Range of  $i \in [1, N]$

$$a \% b == 0$$

Factor, if ' $b$ ' is factor of ' $a$ '  $\Leftrightarrow$   $a$  is factor of  $b$

Multiplication: ' $a$ ' is multiple of ' $b$ '  $\Leftrightarrow$   $a$  is multiplication of  $b$

$$10 \rightarrow 1, \underbrace{2, 5,}_{\text{Count = 4}} 10$$

$$1 \rightarrow$$

$$2 \rightarrow$$

$$3 \rightarrow$$

$$\vdots$$

$$!$$

$$10 \rightarrow$$

$$12 \rightarrow 1, \underbrace{2, 3, 4, 6,}_{\text{Count = 6}} 12$$

```
int countOfFactor (int N) {
```

N=10

```
    int count = 0;
```

```
    for (int i = 1; i <= N; i++) {
```

if  $N \div i == 0$  {

// i is factor of N

count++;

}

}

```
return count;
```

Number of iteration: N times

3

$\infty$  times. for N iteration

$\uparrow n$  time ?? time will increase

Unitary Method

5 Apples = 100 Rupees.

1 Apple =  $\frac{100}{5}$  20.

6 Apples =  $20 \times 6$   
 $= 120$  Rupees.

Assumption:  $10^8$  iteration = 1 sec

time for  $(10^9)$  iteration ??

$10^8$  iteration = 1 sec.

$$\frac{8}{4} = \frac{\textcircled{3}}{\textcircled{3}}$$

1 iteration =  $\frac{1}{10^8}$  sec.

$$= 2^{8-2} = 2^1 \textcircled{2}$$

$$\begin{aligned}10^9 \text{ itr} &= \frac{1}{10^8} \times 10^9 \\&= 10^{9-8} = 10 \text{ sec.}\end{aligned}$$

## Improvisation in factor count

N	generation	time	
$10^8$	$10^8$	1 sec.	$10^8 \text{ gen} = 1 \text{ sec.}$
$10^9$	$10^9$	1 sec	$1 \text{ gen} = \frac{1}{10^8} \text{ sec}$
$10^{18}$	$10^{18}$	$10 \text{ sec}$	$10^{18} \text{ gen} = \frac{1}{10^8} \times 10^{18} \text{ sec}$
$\underline{\underline{10^{18}}}$	$\underline{\underline{10^{18}}}$	$\underline{\underline{317 \text{ yrs}}}$	$= 10^{18-8} = 10^{10} \text{ sec.}$

$\frac{N}{10^{18}}$       generation:  $(\sqrt{N})^{1/2}$        $2-4$  rebirth.

$$\sqrt{10^{18}} = (10^{18})^{1/2} = 10^{\frac{18+1}{2}} = (10^9) = 10^{\text{sec.}} \quad (\sqrt{a})^{1/2} = (a^b)^{1/2} = a^{b \cdot \frac{1}{2}}$$

improvisation: we have 2 numbers,  $i, j, N$

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

$$3 * 4 = 12$$

3 & 4 both one factor of 12.

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

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

$$6 * 8 = 24 * 2 \Rightarrow \underline{\underline{48}} = \underline{\underline{48}}$$

$$\underline{\underline{N=100}}$$

$\overset{\circ}{i}$	$N/\overset{\circ}{i}$
1	24
2	12
3	8
4	6
6	4
8	3
12	2
24	1

Point 1:  $\overset{\circ}{i} = 1$       first factor of  $N$

Point 2:  $\overset{\circ}{i} = 2, 3, 4, 6, 8, 12$

observation:

All factors are available  
in point 1.

Point 1:

$$\frac{i}{1} \leq \frac{N}{i}$$

$$\Rightarrow \overset{\circ}{i} * \overset{\circ}{i} \leq N$$

$$\Rightarrow \overset{\circ}{i}^2 \leq N$$

taking square root both sides  
 $\sqrt{i^2} \leq \sqrt{N} \Rightarrow$

$\overset{\circ}{i}$	$N/\overset{\circ}{i}$
1	100
2	50
4	25
5	20
10	10
20	5
25	4
50	2
100	1

Point 1

Point 2

$$\overset{\circ}{i} \leq \sqrt{N}$$

Observation:

1. All the factors are available in first part  $\sqrt{4*4} = \sqrt{4^2} = 4$
2. Range of first part is 1 to  $i \leq \sqrt{N}$ .  $\sqrt{16} \approx 3.87 \approx 4 \equiv ④$

int count\_factor(int N) {

    int count=0;

    for(int i=1; i <=  $\sqrt{N}$ ; i++) {

        if(N % i == 0) {

            // i is factor,  $\frac{N}{i}$  is also a factor

            count+=2;

    }

    return count;

}

$N=16$

$$\sqrt{16} = 4$$

$i$	$N/i$	Count	
1 ✓	16	2	{1, 16}
2 ✓	8	4	{2, 8}
3 ✗			
4 ✓	4	6	[1, 16] [2, 8]

(4) -> loop stop  
count = 4

loop will stop  $\rightarrow ⑤$

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

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

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

by 1 value

```

int count_factor(int N) {
    int count=0; // Number of factors
    for(int i=1; i<=sqrt(N); i++) {
        if(N % i == 0) {
            // i is factor of N
            if(i == N/i) {
                count += 1;
            } else {
                count += 2;
            }
        }
    }
    return count;
}

```

$$N = 100$$

$$\sqrt{100} = 10$$

i	N/i	count	
1	100	2	[1, 100]
2	50	4	[2, 50]
3	33		
4	25	6	[4, 25]
5	20	8	[5, 20]
6	16		
7	14		
8	12		
9	10	9	[10]

loop will stop

→ ⑪

total count of factor = 9

Iteration #:  $\sqrt{N}$

$i \leq \sqrt{N}$   
 square both  
 $i * i \leq \sqrt{N} * \sqrt{N}$   
 $i^2 \leq N$   
 $i \leq N$

## - Prime Number

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

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

Numbers :	15	12	9	23	⇒ prime number
	=	=	=	=	
1	1	1	1	1	
2	2	2	2	2	
3	3	3	3	3	
4	4	4	4	4	
5	5	5	5	5	
6	6	6	6	6	
			③	②	
			⑥		
		12			
count =	④				
			⑤		

Optimised approach.

- Steps:
- ① Calculate number of factors for  $N$ .
  - ② If number of factors == 2, return true.
  - ③ Otherwise return false.

factor count for  $N \Rightarrow \sqrt{N}$  factors.

$\left. \begin{array}{l} \downarrow \text{to } \sqrt{N} \rightarrow \underline{\text{iteration}} \\ \text{scope of iteration} \\ \rightarrow \text{if factor count } > 2 \\ \text{break the loop.} \end{array} \right\}$

```

boolean isPrime (int N) {
    int count = 0;
    for (int i=1; i*i <= N; i++) {
        if (N % i == 0) {
            if (i == N/i) {
                count++;
            } else {
                count += 2;
            }
        }
        if (count > 2) {
            break;
        }
    }
    if (count == 2) {
        return true;
    } else {
        return false;
    }
}

```

$N = 2$   
 $count = \underline{2} \rightarrow \text{true}$

$N = 7$   
 $count = \underline{2} \rightarrow \text{true}$

$N = 9$   
 $count = \underline{2} \rightarrow \text{false}$

$\sqrt{N}$ : iteration.

3

Important stuff about Big-O:

→ not write code on IDE

→ lecture duration: 2.5 hrs

+ 15 minute buffer

→ ~45 min. [Excluding doubt session]

→ About discuss on Syntax part

→ notes one available  
 → Assignment over same Homework hand: publishing Syntax part

- 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.

$A = \begin{array}{|c|c|c|c|c|c|} \hline 10 & 20 & 30 & 40 & 50 & 60 \\ \hline 0 & 1 & 2 & 3 & 4 & 5 \\ \hline \end{array} \Rightarrow \begin{array}{|c|c|c|c|c|c|} \hline 60 & 50 & 40 & 30 & 20 & 10 \\ \hline 0 & 1 & 2 & 3 & 4 & 5 \\ \hline \end{array}$

given an array and two indices i1 and i2.

swap the elements of i1 index and i2 index.

$A = \begin{array}{|c|c|c|c|c|c|} \hline 10 & 20 & 30 & 40 & 50 & 60 \\ \hline 0 & 1 & 2 & 3 & 4 & 5 \\ \hline \end{array}$        $\underline{i1=1}$   
                         $\underline{i2=4}$

$\text{temp} = 20$

$\text{int temp} = A[i1];$

$A[i1] = A[i2]$

$A[i2] = \text{temp}$

i1 - Index and i2 - Index is swapped.

$\begin{array}{|c|c|c|c|c|} \hline 50 & 40 & 20 & 10 \\ \hline 10 & 20 & 30 & 40 & 50 \\ \hline 0 & 1 & 2 & 3 & 4 \\ \hline \end{array}$

$\uparrow\uparrow$   
 $s \leftarrow e$

$$s=10 \quad \left\{ \begin{array}{l} e=14 \\ s \leftarrow e \\ 2 \quad 2 \end{array} \right.$$

Stopping condition.

Condition of iteration

$s < e$

80	70	60	50	40	30	20	10
10	20	30	40	50	60	70	80
0	1	2	3	4	5	6	7

↑      ↑  
s      e

$s = 0 \quad < \quad e = 7$   
 $\cancel{1} \quad < \quad 6$   
 $\cancel{2} \quad < \quad 5$   
 $\cancel{3} \quad < \quad 4$   
**4      9**

Condition of iteration  $s < e$

int[] reverse(int[] A) {

```

int s=0; →
int e= A.length-1; →
while (s < e) {
    // swap A[s] and A[e]
    {int temp = A[s];
     A[s] = A[e];
     A[e] = temp;
    }
    {s++;
     e--;
    }
}
return A;

```

3

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

$\uparrow \downarrow$   
 s    e

odd length  
array.

$s = 0$  true  $e = 4$

$\cancel{1}$  true     $\cancel{2}$   
 $2$  false     $\underline{\underline{2}}$  → loop will stop.

Even length array

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

↑      ↑  
e      s

$s = 0$  true  $e = 5$

$\cancel{1}$  true     $\cancel{2}$   
 $\cancel{2}$  true     $\cancel{3}$   
 $3$  false     $\underline{\underline{2}}$

- Reverse part of an array

Problem 4: Given an array  $A[]$ , one starting index 'a' and ending index 'b', reverse array block  $\underline{[a \text{ to } b]}$

	$a = 1$ $b = 5$ $s = a$ $e = b$ $= 3$ $\cancel{2}$ $\cancel{1}$ $3$ $\cancel{4}$ $\cancel{5}$ $3$ <i>Stoping condition</i>
$\text{int}[] \text{ reverse}(\text{int}[] A, \text{int } s, \text{int } e) \{$ <pre style="margin-left: 40px;">while (s &lt; e) {     // swap A[s] and A[e]     { int temp = A[s];       A[s] = A[e];       A[e] = temp;     }     { s++;       e--;     } } return A;</pre>	 $60 \ 50 \ 40 \ 30$ $10 \ 20 \ 30 \ 40 \ 50 \ 60$ $\uparrow \quad \uparrow$ $e \quad s$ $s = 2 \ \text{true} \ e = 5$ $2 \ \text{true} \ 4$ $4 \ \text{false} \ 3 \rightarrow \text{loop will stop.}$

## - Rotate an array

Google, Amazon  
Problem 5: Given an array  $A[]$  and int value  $K$ , rotate array  $K$ -times from last to first.

Example 1:  
 $A = \begin{matrix} 10 & 20 & 30 & 40 & 50 & 60 \end{matrix}$

$K=2$

1<sup>st</sup> rotation  $\Rightarrow \begin{matrix} 60 & 10 & 20 & 30 & 40 & 50 \end{matrix}$

2<sup>nd</sup> rotation  $\Rightarrow \begin{matrix} 50 & 60 & 10 & 20 & 30 & 40 \end{matrix}$

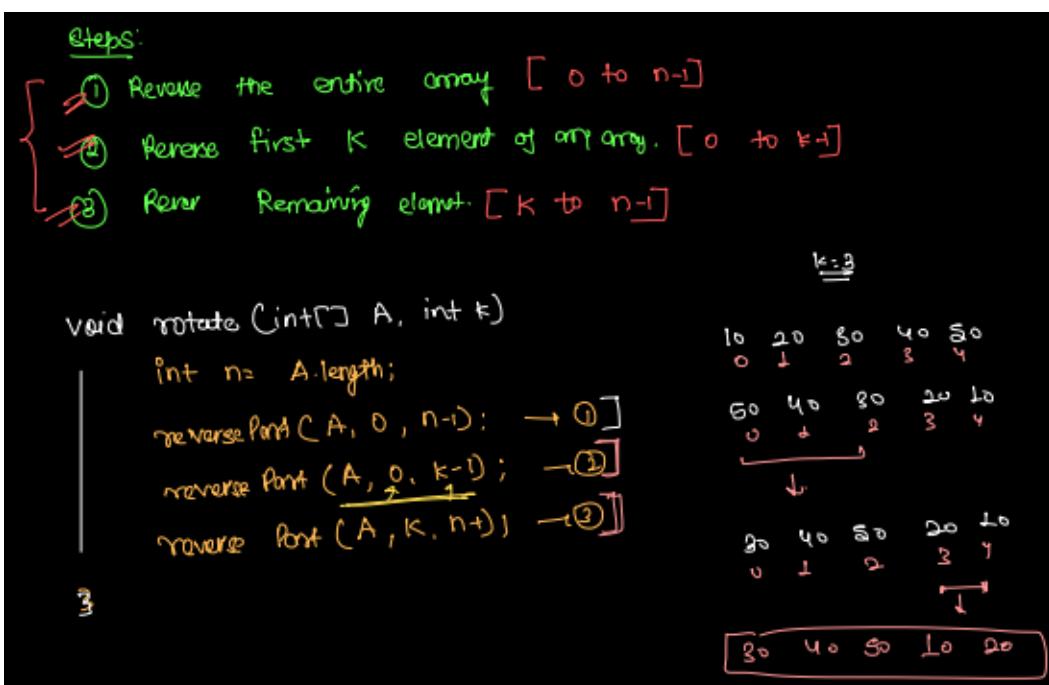
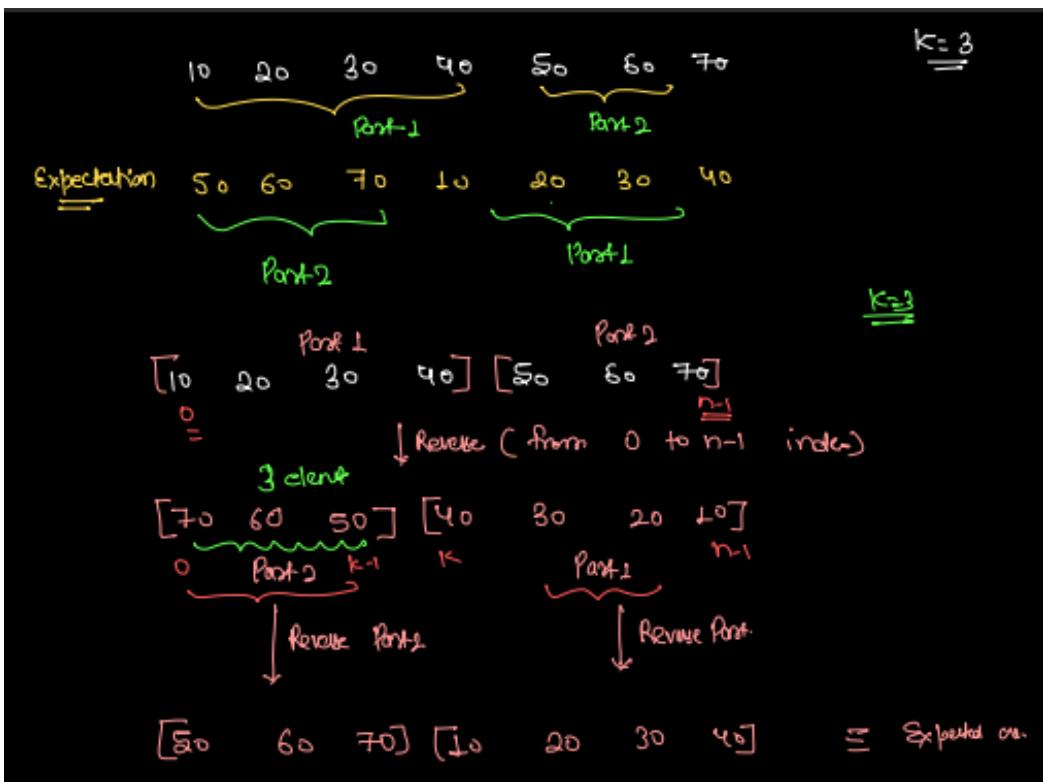
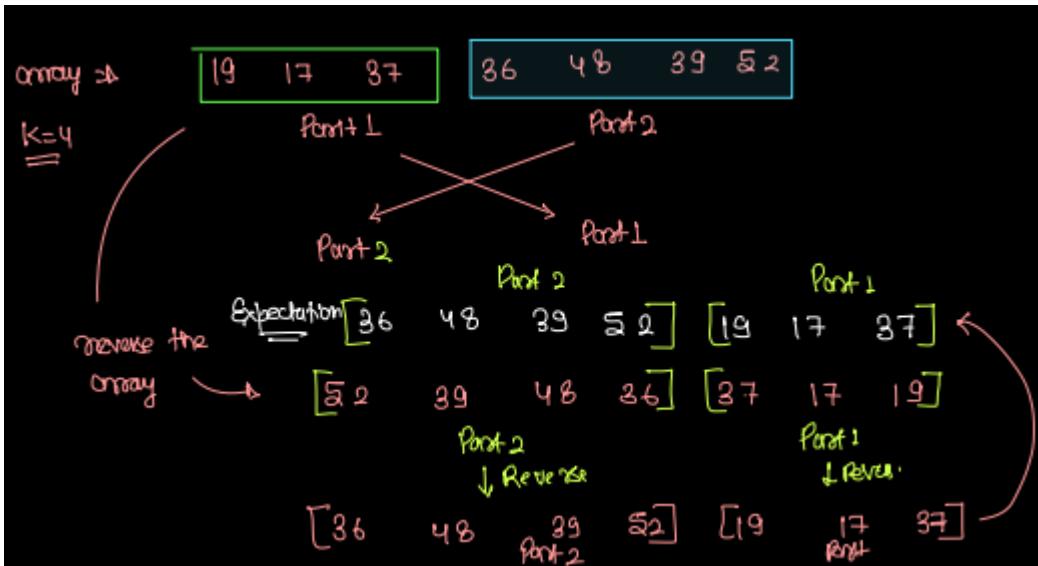
3<sup>rd</sup> rotation  $\Rightarrow \begin{matrix} 40 & 50 & 60 & 10 & 20 & 30 \end{matrix}$  ] final answer

Example 2:

array  $\Rightarrow \begin{matrix} 19 & 17 & 37 \end{matrix} \quad \begin{matrix} 36 & 48 & 39 & 52 \end{matrix}$

$K=4$

$\begin{matrix} 36 & 48 & 39 & 52 \end{matrix} \quad \begin{matrix} 19 & 17 & 37 \end{matrix}$



10 20 30 40  
0 1 2 3

K=6



$$n=4 \quad 1 - 5$$

$$\sqrt[4]{\cdot} \cdot 4 = 1$$

$$\sqrt[4]{\cdot} \cdot 4 = 2$$

$$\sqrt[4]{\cdot} \cdot 4 = 3$$

if length is n:

K = K % n → Rotated k  
in range 0 to n-1

void rotate (int A[], int k)

int n = A.length;

K = K % n; ] → net rotation

n=7  
k=3

reversePart (A, 0, n-1);

reversePart (A, 0, k-1);

3 % 7 = 3

reversePart (A, k, n-1);

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

$\log_b a = ?? \rightarrow b$  power what is equal to a ??

$$\log_b a = c$$

$$\Rightarrow b^c = a$$

$\log_2 64 = ?$	$2^? = 64$	$\log_{10} 10000 = ?$	$10^? = 10000$
$\log_2 64 = 6$	$2^6 = 64$	$\log_{10} 10000 = 4$	$10^4 = 10000$
$\log_2 8 = ?$	$2^? = 8$	$\log_3 81 = ?$	$3^? = 81$
$\log_2 8 = 3$	$2^3 = 8$	$\log_3 81 = 4$	$3^3 = 3 \times 3 \times 3$ $3^3 = 81$

$$\log_3 27 = x \quad 3^x = \boxed{27}$$

$\log_3 27 = 3$

$$x = ? \quad 3^x = 3 \times 3 \times 3$$

$$3^x = 27$$

$$\boxed{x=3}$$

$$\log_2 32 = ?? \quad 2^? = 32 \quad 2^5 = \frac{2 \times 2 \times 2 \times 2}{=} \\ ? = 2 \times 16 \quad = 32$$

$\log_2 32 = 5 \cdot \cancel{xyz} \equiv \boxed{5}$

Properties:

$$\textcircled{*} \quad N = 2^K$$

$$\Rightarrow K = \log_2 N$$

$$\textcircled{*} \quad \log_b b^N = \boxed{N}$$

$$\log_b b^N = ?$$

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

Problem Solving & DSA		
<u>- Time Complexity</u>		
- Arrays [prefix sum, subarray, sliding window, 2D Matrices]	{ Data Structure }	
- Bit Manipulation		
- Hashing		
- Recursion		
- Sorting		
- Searching		
- 2D Pointer Technique		
- Strings		
- LinkedList		
- Trees, BST, Heap		
- Dynamic Programming		
- Graph		

} techniques + Algorithms

} D.S.

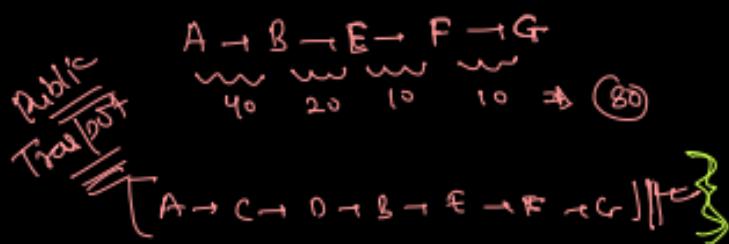
22 sessions  $\equiv$  4 months

DSA  $\Rightarrow$  Data Structure +

- { tree
- graph
- array
- linked list
- Stack
- Queue

Algorithm:

- { kadane's algo
- dijkstra's
- kruskal
- bellman ford



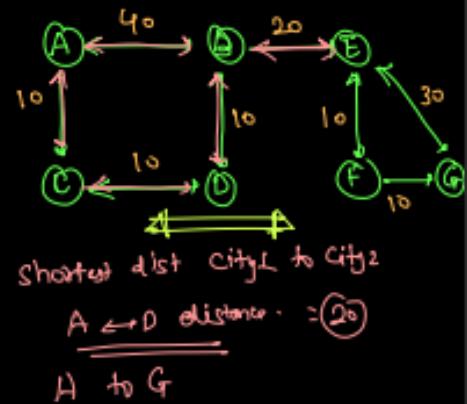
By Post

Graph.

Shortest distance  $\rightarrow$  City encounter is min

A to G

A to B to E to G }

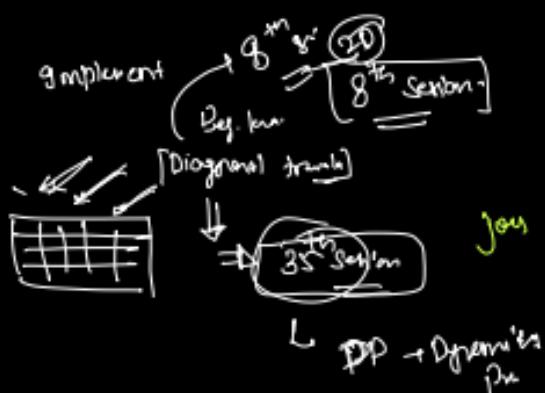


google map

{ Dijkstra  
Kruskal

Hard work  $\rightarrow$  Revision +  
Dry Run

Problems  $\Rightarrow$  for problems  
Scatter is sufficient.



you  $\rightarrow$  struggle

Chapter

A

Chapter 21

game

[25%]

# 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.

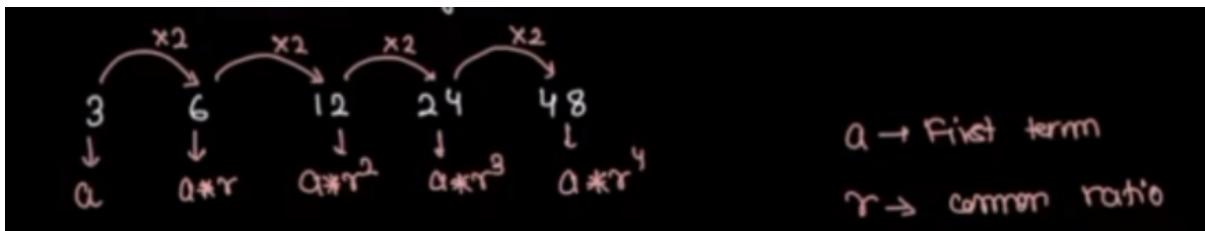
$$S_n = \frac{N(N+1)}{2}$$

## - 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

$r / a = \text{common ratio}$  - 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

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

### 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

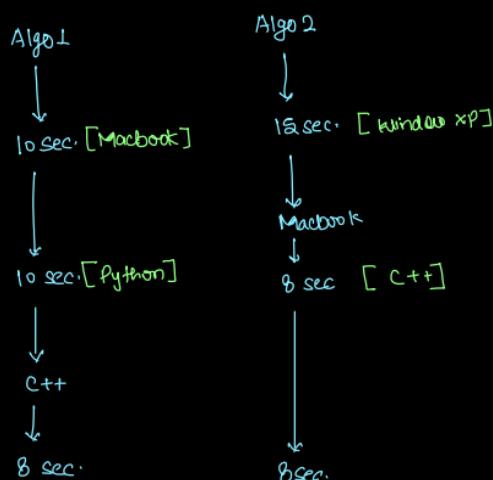
- Comparison Based on execution time
  - Comparison based on iteration and graphs
  - How to calculate Big-O notation
  - Why neglect lower order term?
  - Why neglect constant coefficient ?
  - issue in big-O notation
  - Space complexity
  - Time Limit Exceed
- 
- **Comparison Based on execution time**

Execution time: code start at time  $t_1$  and end at time  $t_2$   
difference b/w time is known execution time.

$$\text{Execution time: } t_2 - t_1$$

Comparison of Algorithms using Execution time:

**NOTE:** Both algo's have same size of input  $N$



## - Comparison based on iteration and graphs

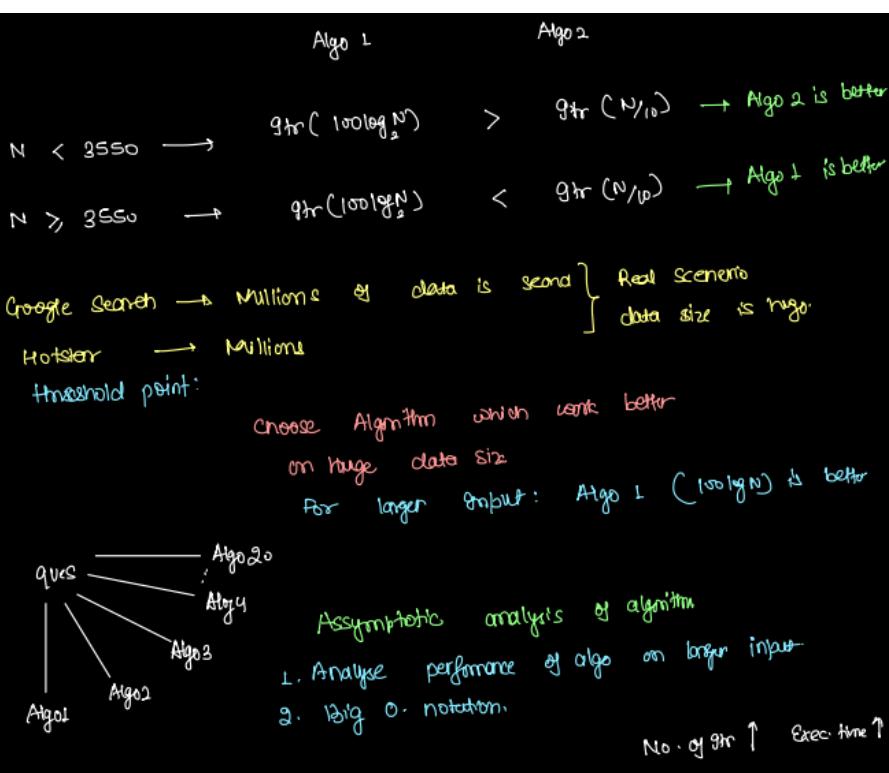
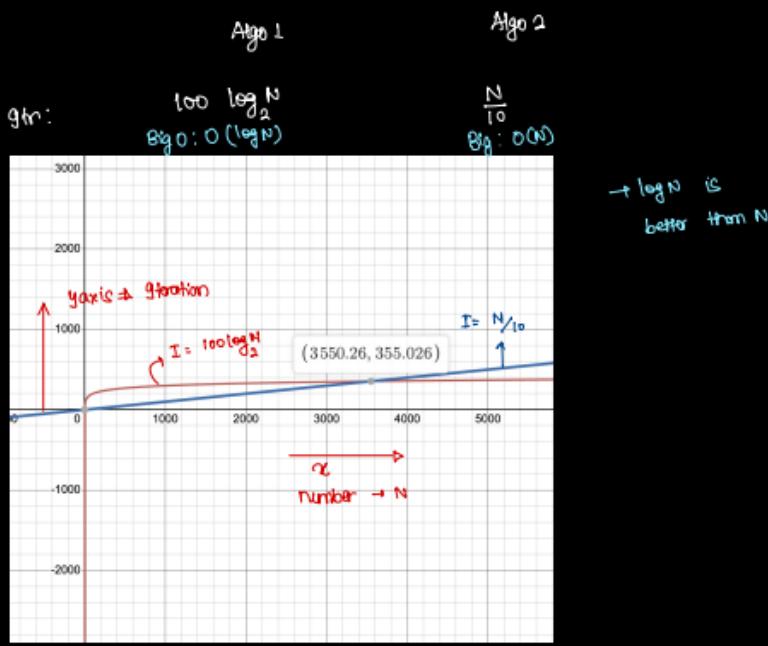
Conclusion: Comparing of two algs based on their execution time is not good option because exec. time depends on lot of external factors.  
 [Ex: machine config, language, environment etc]

```
for (int i = 1; i < N; i++) {
    S(i);
}
```

Macbook → N  
 XP → N  
 Python → N  
 C++ → N  
 Volcomo → N  
 Slackin → N

No impact of machine, surrounding and lang on number of iterations

Comparison of two Algs based on iteration:



## - How to calculate Big-O notation

How to find Big-O notation?

① calculate total number of iteration.

② Neglect lower order terms [keep highest term only]

③ Neglect constant coefficient

$$f(n) = \underbrace{3n^2 + 6n + 10^3}_{\text{No. of iteration}}$$

$$\text{T.C. } O(n^2) \quad \underline{\underline{\text{Ans.}}}$$

$$f(n) = 3n^2 + 6n^3 + 10^3$$

$$\text{T.C. } O(n^3) \quad \underline{\underline{\text{Ans.}}}$$

$\frac{\text{Iteration}}{n} < [n \log n]$

$$f(n) = 3n + 6n \log n + 10^3$$

Algo1      Algo2

Algo1 is better.

$$\text{T.C. } O(n \log n)$$

$$f(n) = 10^3 \quad \left. \begin{array}{l} \text{No. of Iteration is constant} \\ \text{and independent of N.} \end{array} \right\}$$

$$\text{T.C.} = O(1) \rightarrow \text{constant}$$

```
int solve (int n) {
    for (int i=1; i<=n; i++) {
        sop(i);
    }
    return 1;
}
```

Relation b/w Execution time and no. of iteration.

No. of iteration  $\uparrow \Rightarrow$  Execution time  $\uparrow$

## - Why neglect lower order term?

why we neglect lower order term ?

$$gtr = \underbrace{n^2}_{\text{High order}} + \underbrace{10n}_{\text{Lower order}}$$

<u>N</u>	<u>total gtr</u>	% of lower order term in total gtr
10	$100 + 100$	$\frac{100}{200} \times 100 = 50\%$
100	$10^4 + 10^2$	$\frac{10^2}{10^4 + 10^2} \times 100 = \frac{10^2}{10^4} \times 100 = 1\%$
1000	$10^6 + 10^4$	$\frac{10^4}{10^6 + 10^4} \times 100 = \frac{10^4}{10^6} \times 100 = 1\%$

Conclusion: For high values of n, contribution by lower order term is very less in total gtration.  
That's why we discard lower order term.

## - Why neglect constant coefficient ?

why we neglect constant coefficient ?

	Algo 1	Algo 2
Better ?	$2n^2$	$\frac{5n}{2}$
	$O(n^2)$	$O(n)$
	More gtration than Algo2	→ Better, no. of gtration is less

Conclusion: for larger input, constant coefficient will not play significant role.

Algo 1	$2n^2$	$7n^2$	$6n^2$
$n=1$	2	7	6
$n=100$	$3 \times 10^4$	$7 \times 10^4$	$6 \times 10^4$
$n=10000$	$3 \times 10^8$	$7 \times 10^8$	$6 \times 10^8$

Algo 2	$5n$	$2n$	$6n$
$n=1$	5	2	6
$n=100$	$5 \times 10^2$	$2 \times 10^2$	$6 \times 10^2$
$n=100000$	$5 \times 10^5$	$2 \times 10^5$	$6 \times 10^5$

- issue in big-O notation

Some issues in big-O notation:

I.	Algo 1	Algo 2
gtr:	$10n$	$n^2$
Big O:	$O(n)$	$O(n^2)$

Claim: Algo 1 is always better?  $\times$

Value of N	gtr in Algo 1	gtr in Algo 2	which one is better?
5	50	25	Algo 2 is better
8	80	64	Algo 2 is better
10	100	100	Both are same
11	110	121	Algo 1 is better

for  $N \geq 11 \rightarrow$  Algo 1 is better

Algo 1 is better than Algo 2 after some certain value [threshold value]

Algo 1	Algo 2	
gtr:	$2n^2 + 7n$	$5n^2$
Big O:	$O(n^2)$	$O(n^2)$

Algo 1	Algo 2
$2n^2 + 7n$	$5n^2$
$n(2n+7)$	$n(5n)$
$A * B$	$A * C$
Relation $B \leq A \leq C$	
$2n+7 < 5n$	
$\hookrightarrow$ After some threshold $5n$ , $C$ is greater.	
$gtr (2n^2 + 7n) < 5n^2$	
$\rightarrow$ Algo 1 is better than Algo 2	

2.

```

boolean search (int [] A, int data) {
    for (int i=0; i < A.length; i++) {
        if (A[i] == data) {
            return true;
        }
    }
    return false;
}

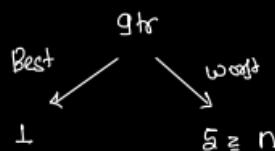
```

3

20	50	10	40	20
0	1	2	3	4

gtr.

$\text{data} = 20 \rightarrow 1$   
 $\text{data} = 40 \rightarrow 4 \quad \} \text{true}$   
 $\text{data} = 20 \rightarrow 5$   
 $\text{data} = 99 \rightarrow 5 \quad \} \text{false}$   
 $\text{data} = 100 \rightarrow 5$



NOTE: with BigO notation  $\rightarrow$  we always deal with worst scenario

ptr: n

$\Rightarrow O(n)$

8:32 — 8:42  
Break time.

## - Space complexity

space complexity:

```

void fun (int n) {

```

```

    int x=10;
    long l=25.6L;
    int z=23;
}

```

3

int  $\rightarrow$  4 byte  
long  $\rightarrow$  8 byte

Total space =  $n \rightarrow 4$  bytes  
 $x \rightarrow 4$  bytes  
 $l \rightarrow 8$  bytes  
 $z \rightarrow 4$  bytes  
 $= 20$  bytes

```

void fun(int n) {

```

```

    int x=10;
    int [] A = new int[n];
    int [][] B = new int[7][n];
}

```

3

Total space

$n \rightarrow 4$  byte  
 $x \rightarrow 4$  byte  
 $A \rightarrow n * \text{int} \Rightarrow n * 4$  byte  
 $B \rightarrow 7 * n * \text{int} \Rightarrow 7n * 4$  byte

total space =  $(4 + 4 + 4n + 28n)$  byte

=  $32n + 8$  bytes

How to calculate space complexity for an algorithm.

Input  $\rightarrow \{ \text{Alg} \} \rightarrow \text{return ans/op.}$

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

N = size of array.		
T.C.	S.C.	
gtr: N	space complexity	
T.C.: $O(n)$	space: 8 byte	
	S.C. = $O(1)$	
	constant size	

```

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

```

N = A.length		
T.C.	S.C.	
gtr: n	Space = 8 byte	
T.C.: $O(n)$	S.C. : $O(1)$	

```

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

```

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

T.C.			S.C.
gtr: n			Space
$O(n)$			$n \rightarrow \text{int} = 4 \text{ byte}$
			$8 \rightarrow n \times \text{int} = n \times 4 \text{ byte}$
			<del>2 int</del> $\rightarrow \text{int} = 4 \text{ byte}$
			$i \rightarrow \text{int} = 4 \text{ byte}$
			Total Space = $4 + 8n + 8 + 4$
			$= 4n + 16 \text{ bytes}$
			Big O: $O(n)$

```

int solve(int[] A, int k) {
    int n = A.length;
    int[] B = new int[n];
    for (int i = 0; i < n; i++) {
        B[i] = A[i];
    }
    return B[k];
}

```

```
int solve(int[] A, int val) {
```

- ✓  $\text{int } n = A.length;$
- ✓  $\text{int[]} arr1 = \text{new int}[2*n];$
- ✓  $\text{int } n1 = A[0];$
- ✓  $\text{int } n2 = A[1];$
- ✓  $\text{for } (\text{int } i=n; i > 0; i=i/2) \{$

| ✓ 3 int var

3

- ✓  $\text{int}[][] arr2 = \text{new int}[n][n];$
- ✓  $\text{return } arr2[2][0];$

3

Time complexity?

→ Time complexity depend on for loop of code  
 $n = 32 \rightarrow 16 \rightarrow 8 \rightarrow 4 \rightarrow 2$   
 $\downarrow$   
 $\boxed{\log(n)}$   
 $\downarrow$   
 T.C. :  $O(\log n)$

Var	Var count	Space
$arr1$	$2*n$	$(2*n) * 4 \text{ byte}$
$n1$	1	4 byte
$n2$	1	4 byte
$i$	1	4 byte
3 int	2	3 * 4 byte
$arr2$	$n*n$	$(n*n) * 4 \text{ byte}$

$$\text{Total space} = 4 + 2*4*n + 4 + 4 + 4 + \underbrace{3*4}_{\text{in}} + 4n^2$$

$$= (4) + (8n) + (12) + (12) + (4n^2)$$

$$= 4n^2 + 8n + 28 \text{ byte}$$

in Big O notation :  $O(n^2) \rightarrow \text{S.C}$

## - Time Limit Exceed

Time Limit Exceed:

TLE: [language] → Mistake in loop → infinite time  $\rightarrow$  TLE

Beg. DSA → count factor → simple Approach → TLE

$\downarrow$   
Optimise Approach → working fine ↗

online IDE/  
platform → server → processing speed  
 $\sim$   
restriction  
 $10^8 \text{ gtoe} = 1 \text{ sec.}$

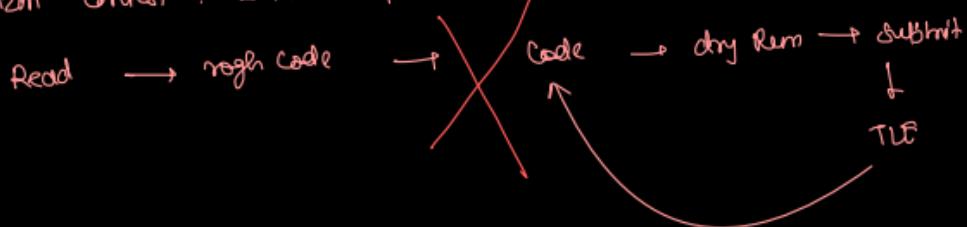
→ For per user → server gives  $\frac{1}{10^8}$  sec.

num code within 1 sec

$\rightarrow 10^8 \text{ gtoe} \rightarrow \text{TLE}$

Our code should have at max  
Solve 800  $\rightarrow 10^7$  to  $10^8$  gtrs is the

Amazon Contest : Lim. 2 prob's.



constraint:

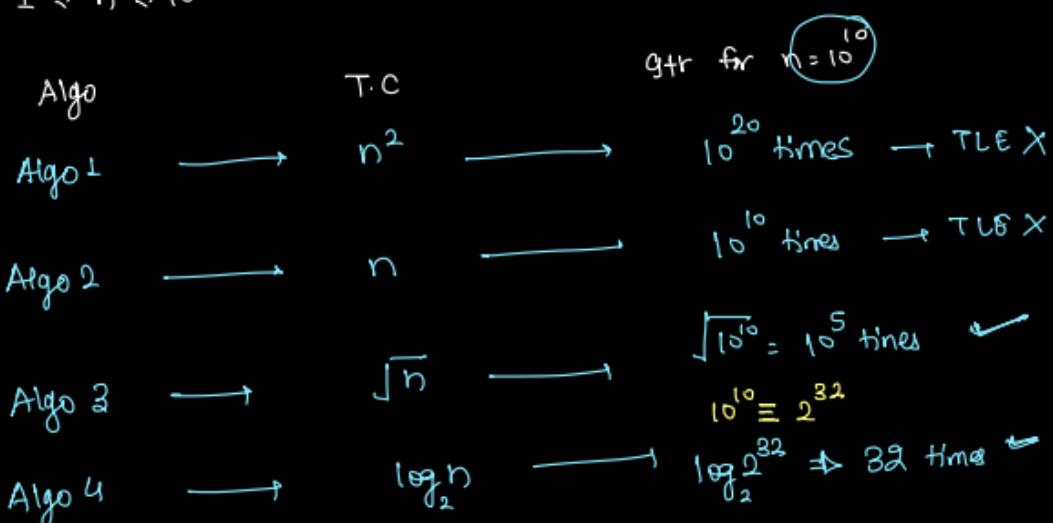
$$1 \leq N \leq 10^5 \quad \text{max possible value of } N \rightarrow 10^5$$

Rough code  $\rightarrow N^2$  gtrs      if  $n = 10^5$   
 $\downarrow$                           no. of itr  $= (10^5)^2 = 10^{10}$   
 don't wait                          TLE.  
 for TLE:  
 $\downarrow$   
 Improve  
 your logic for make  
 Iteration  $< 10^8$

Count Factor in Beg. Long.	Count Factor in Beg. DSA
$1 \leq N \leq 200$ $1 \leq N \leq 1000$ Brute force $T.C. O(n)$ $\therefore$ Max possible gtrs are $10^8$ $1 \text{ sec} = 10^8 \text{ gtrs}$ $10^8 \text{ gtrs will run within}$ $1 \text{ sec}$ .	$1 \leq A \leq 10^9$ Brute force, T.C. $O(n)$ Max iteration $\rightarrow 10^9$ $1 \text{ sec} = 10^8 \rightarrow$ $10^9$ sec is required. $\times$ $\rightarrow$ TLE with $O(n)$ Approach optimised Approach $T.C. O(\sqrt{n})$ $\sqrt{10^{10}} = 10^5$ $1 \text{ sec} = 10^8 \text{ gtrs}$ $10^5 \text{ gtrs is in range of 1 sec}$ $\rightarrow$ gtr will work fine

Problem  
Constraint

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



-	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  $\rightarrow$  Rejt  $\rightarrow$  ]  $\rightarrow$  concept  
 $\rightarrow$  cranking  
 $\rightarrow$  spoon feeding

$\log n \rightarrow$   
 $\downarrow$   
 Brute force  
 $\square$   
 Normal

Brute force

&  
 figure out issue

&  
 trigger about logic

$\downarrow$   
 optimise approach.

Solution  $\rightarrow \log n$   
 $\downarrow$

Brute force  $\times$

$$100 = 2^7$$

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

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

Equivalent  
 $32 \equiv 32$   
 $\square$   
 somehow

$$\begin{aligned} 10^{10} &\equiv 2^{35} \equiv 2^{22} \equiv 2^{21} \equiv 10 \equiv 9 \\ &10 \equiv 11 \\ &\underline{\underline{32 \equiv 32}} \end{aligned}$$

Approximation

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	gtr on I
0	0 to 0	1
1	0 to 1	2+ 1
2	0 to 2	3+ 2+ 1
3	0 to 3	4+ 3+ 2+ 1
...	...	...
n-1	0 to n-1	n

$$\text{total gtr} = 1 + 2 + 3 + 4 + \dots + n$$

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

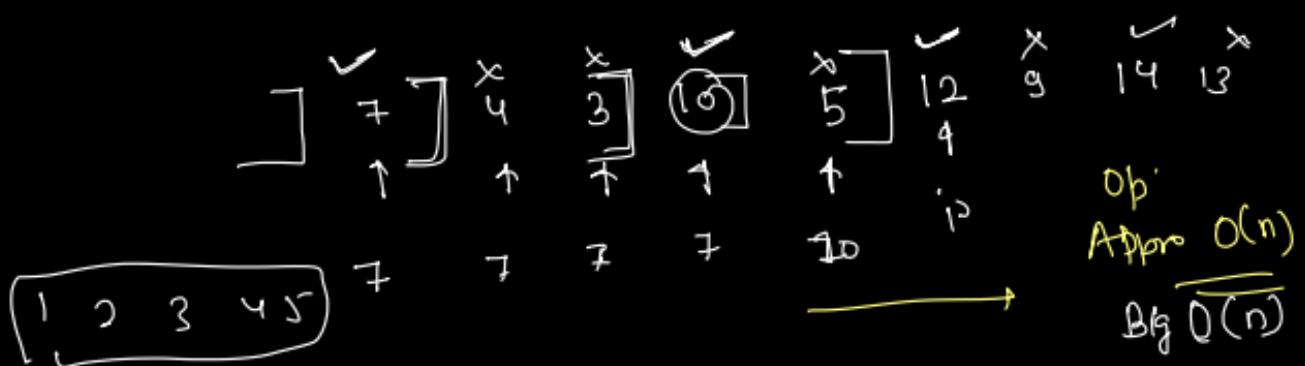
---

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

T.C =  $(n^2)$

2 → 0 to 2 max

Big  $O(n^2)$

3 → 0 to 3 max

4 → 0 to 4 max

⋮

n → 0 to n max

⋮

## 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 of iteration 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

Problem 1: Count Pairs 'a-g'

Given a char array ch[] of size N, calculate number of pairs

indices = i, j such that  $i < j \&& ch[i] == 'a' \&& ch[j] == 'g'$ .

All characters in array are in lowercase.

Constraints :

$1 \leq N \leq 10^5$

'a'  $\leq ch[i] \leq 'z'$

$\Rightarrow ghr = n^2 = 10^5 \times 10^5 = 10^{10}$

1 sec =  $10^8$  ghr

$10^{10}$  will take more than one sec.

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

Ex:	b		a		a		g		g		d		c		a		g
	0		1		2		3		4		5		6		7		



Count pairs = 5

II	$i, j$	$i, j$	$i, j$
$ch[i] == 'a'$	$(1, 3)$	$(2, 3)$	$(6, 7)$
III	$(1, 7)$	$(2, 7)$	$(6, 7)$

Ex 2.

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

I

$i < j$

ch[i] = 'a'

ch[j] = 'g'



(i,j)

(0,2)

(0,4)

(0,6)

(0,3)

(2,4)

(3,6)

(5,6)

total pairs

6 Ans

Ex.

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

I

$i < j$

All pairs

Pair count

II

$i=0$

(0,1)

(0,2)

(0,3)

(0,4)

(0,5)

→ 0

ch[i] = 'a'

$i=1$

(1,2)

(1,3)

(1,4)

(1,5)

→ 3

III

ch[j] = 'g'

$i=2$

(2,3)

(2,4)

(2,5)

→ 0

$i=3$

(3,4)

(3,5)

→ 2

$i=4$

4,5

→ 0

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;

3

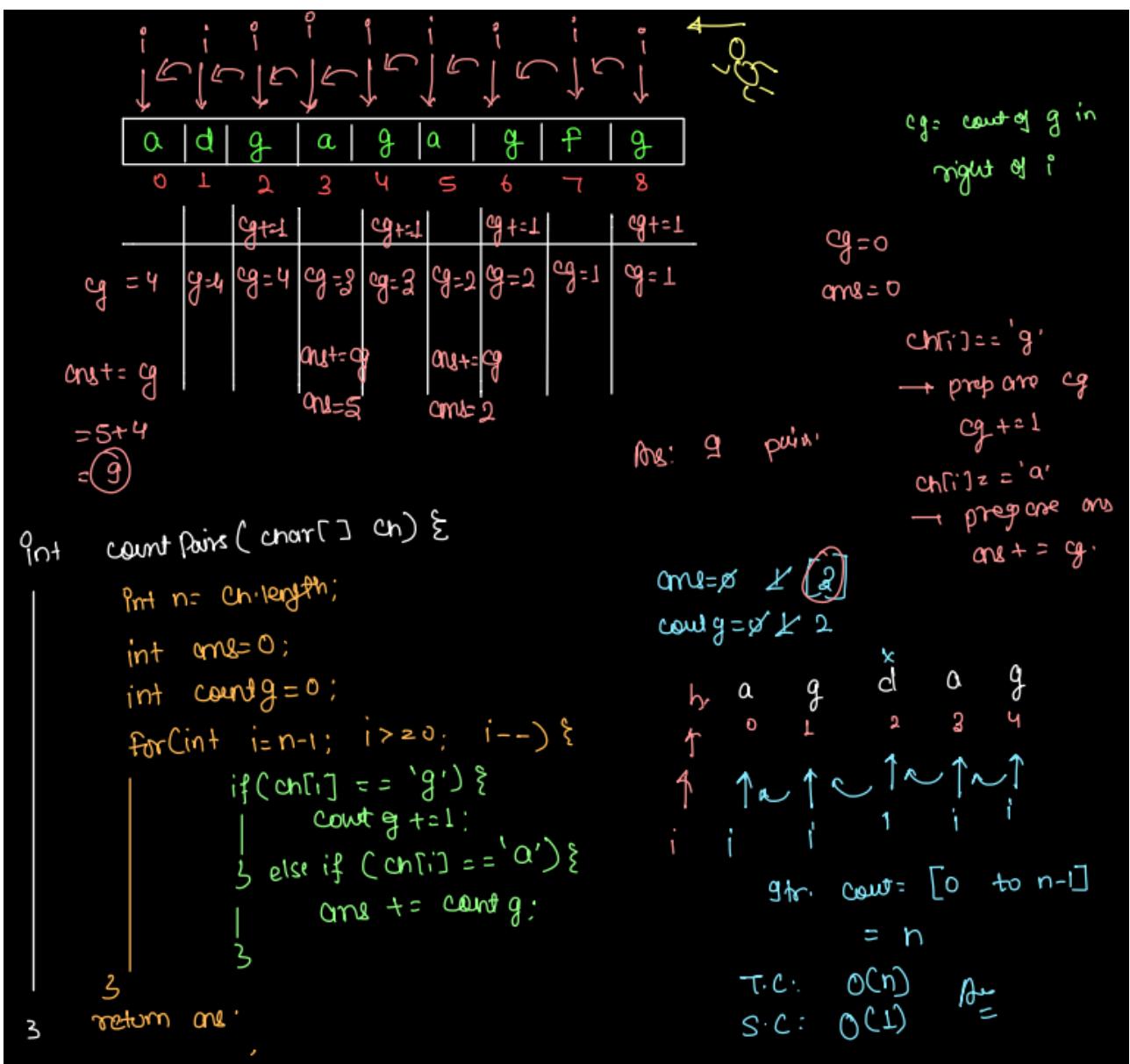
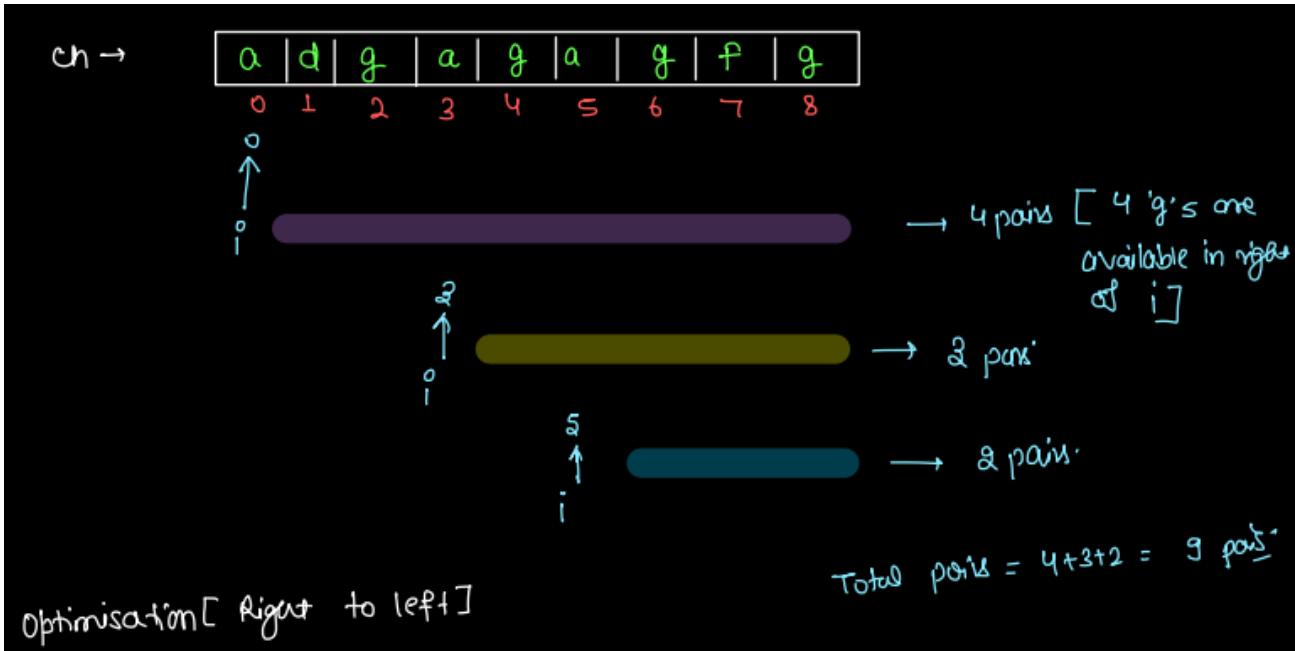
3

Total str = 0 + 1 + 2 + 3 + ... + n-1

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

Value of i	Ranged	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

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



summary: ① Iteration from Right to left

Why?

count of 'g' from  $i+1$  to last index.

Iterate from last to first; we can track the count

of 'g' from  $i+1$  to  $n-1$ .

② If  $ch[i] = 'g'$   $\{ \text{count}_g += 1 \}$

Increase  $gCount$  by 1

③ If  $ch[i] = 'a'$   $\{ ans += \text{count of } g \}$

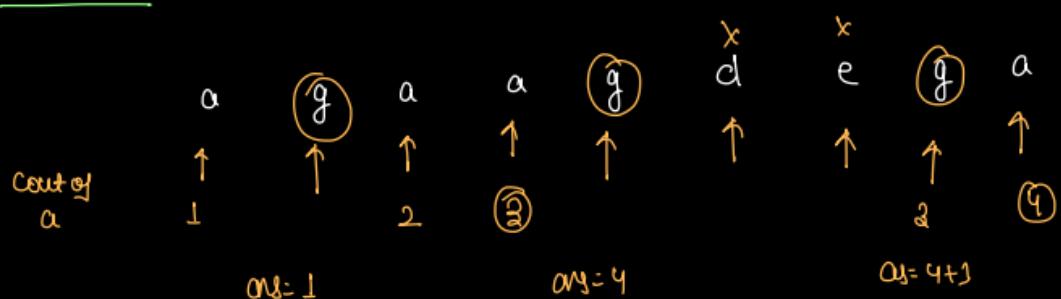
If char at  $i^{th}$  index is 'a' & we have

count of 'g' available in answer

possible pairs are nothing but gCount in  
right side of  $i^{th}$  index

$\rightarrow ans += \text{count of } g$

Method 2: [Iteration from left to Right]



TODO:

$\rightarrow$  generate from left to right

$\rightarrow$  manage count of 'a'

$\rightarrow$  if 'g' encountered prepare your ans

[Break]: 8:11 - 8:25  
time

## - Leaders in an array

### Problem 2: Leaders in an Array

Given an array  $\text{arr}[]$  of size  $N$ , You have to find leaders in  $\text{arr}$ .

Note 1:  $\text{arr}[i]$  is said leader

| if it is greater than max of all elements from left side  $[0, i-1]$

Note 2:  $\text{arr}[0]$  is considered as leader.

constraint: (1)  $1 \leq N \leq 10^5$  →  
(2)  $1 \leq \text{arr}[i] \leq 10^9$

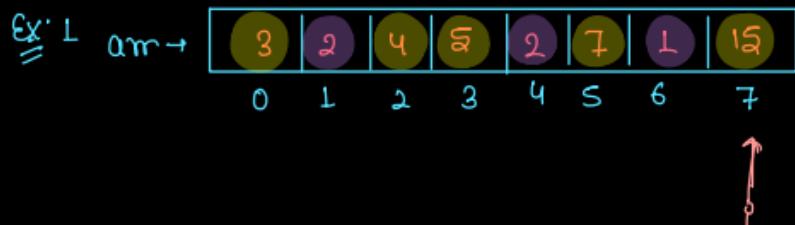
$\text{Max } N = 10^5$

gtr using  $O(n^2)$

$10^5 \times 10^5 = 10^{10}$   
→ 1 sec < time

$O(n^3) \rightarrow \text{TLE}$

calculate leader count?



leaders count = 2

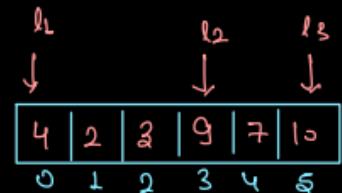
```
int leaderCount(int arr[]) {
    int n = arr.length;
    int count = 1;

    for(int i=1; i<n; i++) {
        // calculate max from 0 to i-1
        int max = arr[0];
        for(int j=1; j<i; j++) {
            if(arr[j] > max) {
                max = arr[j];
            }
        }

        // if arr[i] is greater than max [0:i-1]
        // arr[i] is leader value
        if(arr[i] > max) {
            count++;
        }
    }

    return count;
}
```

$n=6$

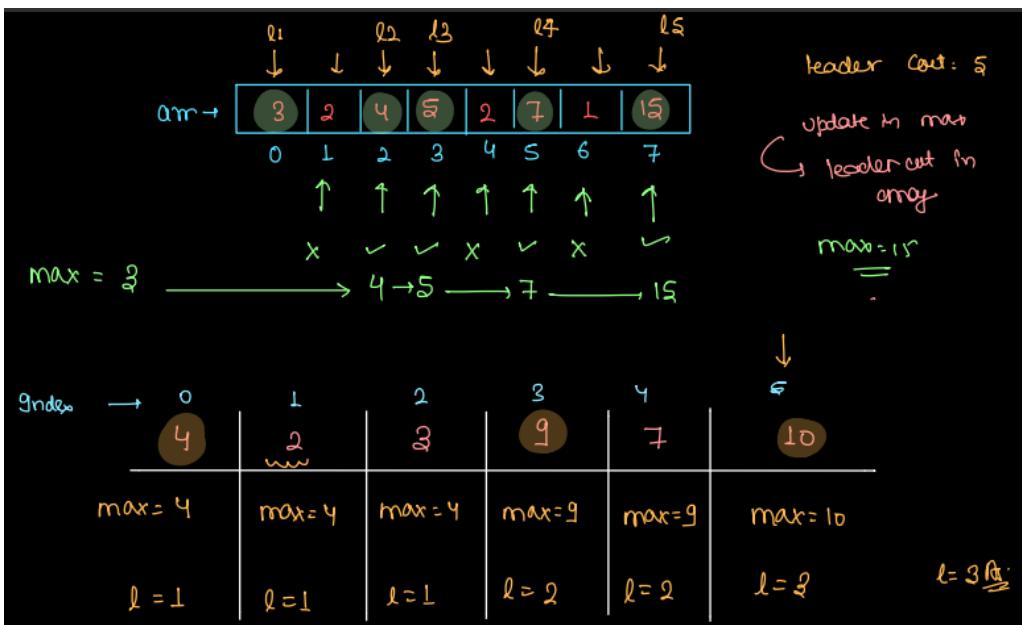


count = 2

$$\text{Total gtr} = 0 + 1 + 2 + 3 + \dots + n-2$$

$\approx O(n^2)$

Value of i	Range of j	Item count
1	[1, 1)	0
2	[1, 2)	1
3	[1, 3)	2
4	[1, 4)	3
$i$	$[1, n-1)$	$n-2$



```
int leaderCount (int[] arr) {
```

```
int n = arr.length;
```

```
int count = 1;
```

`int max = arr[0];`

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

if( arr[i] > max) {

$$\max = 4m$$

1

```
    return cow;
```

T.C : O(n)

$S, C : O(1)$

Revision:

$$\textcircled{1} \quad \begin{matrix} b & a & g & a & g & g \\ 2 & & 2 & & & \end{matrix} = \textcircled{5} \quad 11^{\text{v}}$$

(2)	4	2	3	9	7	10
	0	1	2	3	4	5

→ legal error: 3 A

### - N Bulbs

Prashant B. N. Patil

$\rightarrow \text{DN} \rightarrow \text{arr}[i] = 1$

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 :  $1 \leq N \leq 10^5$

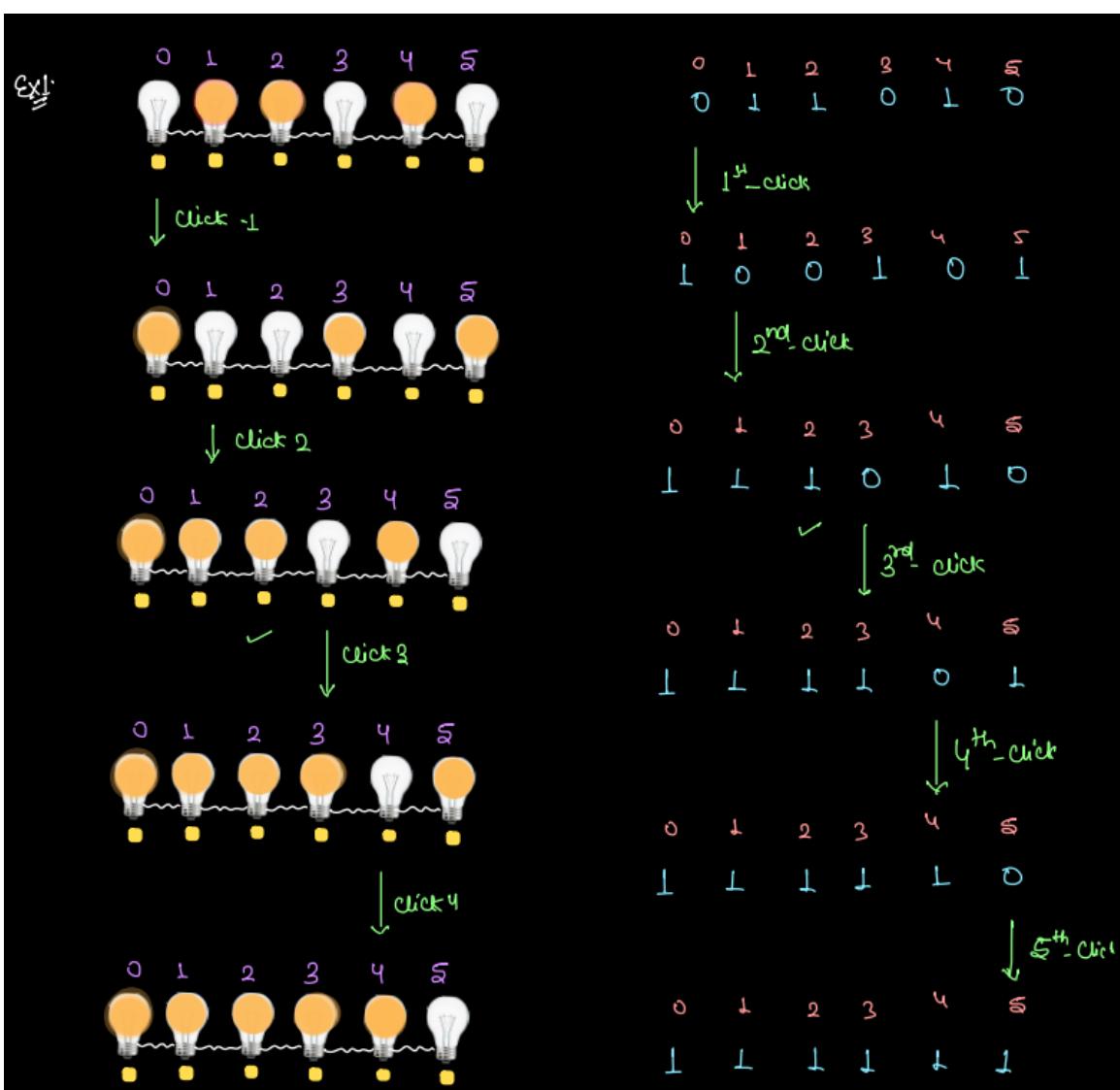
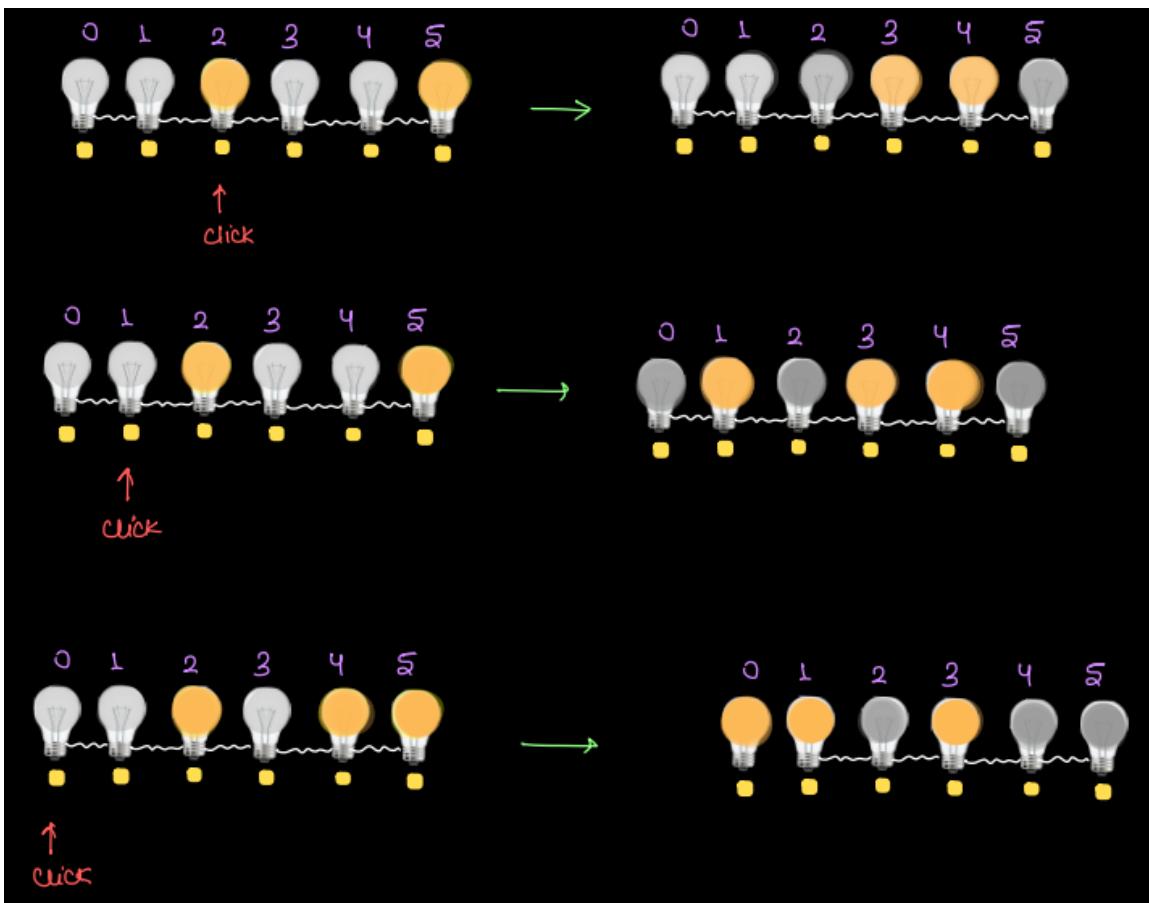
that every bulb will "ON".

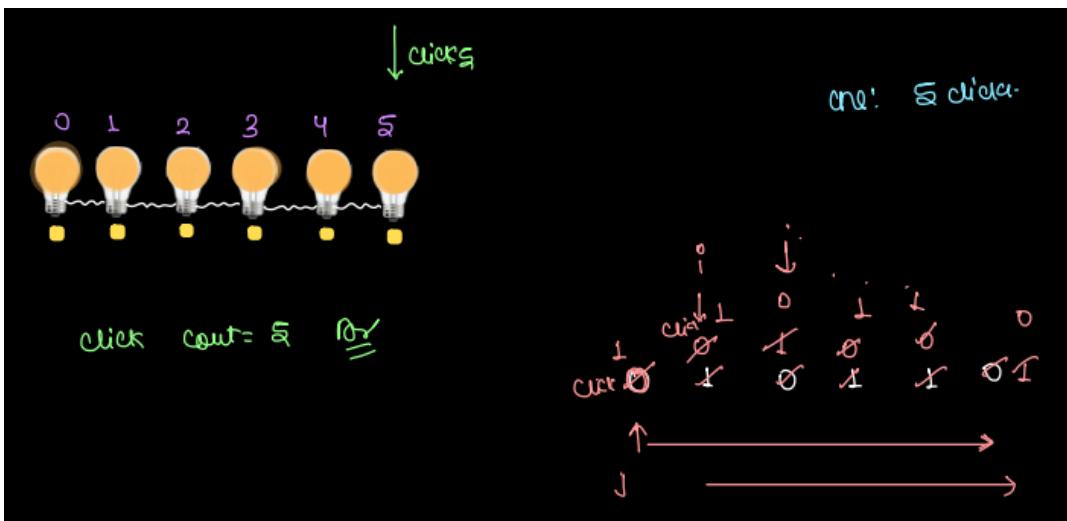
constraints :  $1 \leq N \leq 10^5$        $\hookrightarrow O(n) \Rightarrow \max gtr = 10^5 \rightarrow \text{manageable in } 1/10 \text{ sec}$

$$O(n^2) : 10^5 \times 10^5 = 10^{10} \rightarrow 1 \text{ sec X}$$

$\mathcal{O}(n^2) \rightarrow \text{TLG}$



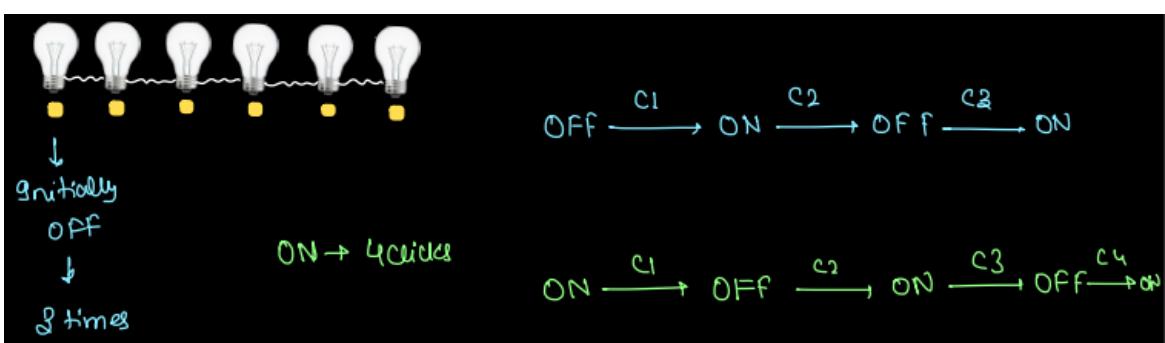




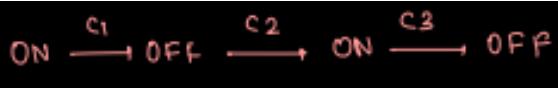
```

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 from 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;
}
    
```

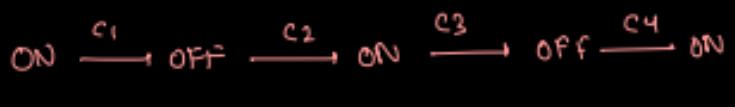
T.C:  $O(n^2)$   
S.C:  $O(1)$



ON → 3 time  
clicks



ON → 4 time  
clicks



OFF → 3 times  
clicks



OFF → 4 fine



#### **Observation:**

1

```

graph TD
    A[Initial State] -- "Even Clicks" --> B[Final State]
    style A fill:none,stroke:none
    style B fill:none,stroke:none
    
```

$$\text{ON} \xrightarrow{c_1} \text{OFF} \xrightarrow{c_2} \text{ON}$$

2

```

graph LR
    A[Initial State] -- "ON" --> B[Odd Clicks]
    A -- "OFF" --> C[Final State]
    B -- "Odd Clicks" --> D[Final State]
    C -- "Even Clicks" --> D
  
```

The diagram illustrates the relationship between initial and final states and the number of clicks required. It shows three main components: 'Initial State' (represented by a wavy line), 'Odd Clicks' (represented by a wavy line), and 'Final State' (represented by a straight line). An arrow labeled 'ON' points from the Initial State to Odd Clicks. Another arrow labeled 'OFF' points from the Initial State to Final State. A third arrow labeled 'Odd Clicks' points from Odd Clicks to Final State. A fourth arrow labeled 'Even Clicks' points from Final State back to Final State.

	0	1	2	3	4	5
Gravitel State	OFF 	OFF 	ON 	OFF 	OFF 	OFF 
click	✓ F 	F 	F 	F 	F 	F 
			OFF 	ON 	ON 	ON 
			F 	F 	F 	F 
	ON 	OFF 	OFF 	OFF 	OFF 	OFF 
			✓ F 	F 	F 	F 

OFF  $\xrightarrow[3\text{p}\mu\mu]{3\text{cltes}}$  ON

OFF  $\xrightarrow{c_1}$  ON  $\xrightarrow{c_2}$  OFF  $\xrightarrow{c_3}$  off

On  $\frac{2}{\pi} \operatorname{Ai}(x) \rightarrow 0$

Initial State	Flip Counts	Final State
ON	odd	OFF
ON	even	ON
OFF	odd	ON
OFF	even	OFF

→ need a click



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

Initial State	Flip Counts	Final State
ON	odd	OFF
ON	even	ON
OFF	odd	ON
OFF	even	OFF

Click Count (1)

n=5

0 1 1 0 1

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; }

initial ON

click count odd

return c;

c = 0 & 2, 2, 1 → all the bulbs on

else if (arr[i] == 0)

initial OFF

if (c % 2 == 0) { c = c + 1; }

click count even

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

$$\begin{array}{cccccc}
 0 & 1 & 1 & 0 & 1 \\
 0 & 1 & 2 & 3 & 4 \\
 \downarrow c_1 \\
 1 & 0 & 0 & 1 & 0 \\
 \downarrow c_2 \\
 1 & 1 & 1 & 0 & 1 \\
 \downarrow c_3 \\
 1 & 1 & 1 & 1 & 0 \\
 \downarrow c_4 \\
 1 & 1 & 1 & 1 & 1
 \end{array}$$

double: neglect lower order term.

$$f(n) = n^2 + \underbrace{10n}_{\text{contribution of } 10n \text{ in } n^2 + 10n} \\
 = \frac{10n}{n^2 + 10n} \times 100$$

7 Apples + 3 oranges  $\Rightarrow$  % of oranges in total fruits.

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

$$\begin{aligned}
 \text{Sum of G.P.} &= \frac{a(r^n - 1)}{r-1} & a=2 \\
 &= \frac{2(2^n - 1)}{2-1} = \frac{2(2^n - 1)}{1} = 2^{n+1} - 2
 \end{aligned}$$

### 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);  
    }  
}
```

$\log n$

time  $\rightarrow n \log n$   
 $O(n \log n)$

### 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);  
    }  
}
```

$\log 100$   
 $n$

$\log 100 \times n$   
constant  
 $= O(n)$

C++      Java      Python

```
for (i = 0; i < N; i++) {  
    for (j = i; j < N; j++) {  
        break; }  $\rightarrow O(1)$   
    }  $[O(n)]$ 
```

# DSA: Prefix Sum - 27 - Apr 2023

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

## - Prefix Sum Array

~~~~~

**Problem 1. Prefix Sum Array**

~~~~~

Given an Array, create and return prefix sum array where  
 $\text{psum}[i] = A[0] + A[1] + A[2] + A[3] + \dots + A[i]$ .

$A = \boxed{2 | 4 | 5 | -3 | 17 | 8}$        $\rightarrow \text{psum} \rightarrow \boxed{2 | 6 | 11 | 8 | 25 | 33}$

$\begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \end{matrix}$        $\begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \end{matrix}$

↑

$A = \boxed{2 | 9 | -3 | 5 | 1}$

$\begin{matrix} 0 & 1 & 2 & 3 & 4 \end{matrix}$

↓

$\text{psum} \rightarrow \boxed{2 | 11 | 8 | 13 | 14}$

$\begin{matrix} 0 & 1 & 2 & 3 & 4 \end{matrix}$

↑

$\text{psum}[i] = \text{sum of elements in } A \text{ from index } 0 \text{ to } i$

Quiz 1: What will be the prefix sum Array?

$$A = [1, 2, 0, 4, -3, 5]$$

$$A \rightarrow \boxed{1 | 2 | 0 | 4 | -3 | 5}$$

$\begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \end{matrix}$

]

$$\text{psum}[i] = A[0] + A[1] + A[2] + \dots + A[i]$$

$$\text{psum} \rightarrow \boxed{1 | 3 | 3 | 7 | 4 | 9}$$

$\begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \end{matrix}$

↑

### Observations:

$$\text{psum}[0] = A[0]$$

$$\text{psum}[1] = \boxed{A[0]} + A[1] \Rightarrow \text{psum}[0] + A[1]$$

$$\text{psum}[2] = \boxed{A[0] + A[1]} + A[2] \Rightarrow \text{psum}[1] + A[2]$$

$$\text{psum}[3] = \boxed{A[0] + A[1] + A[2]} + A[3] \Rightarrow \text{psum}[2] + A[3]$$

$$\boxed{\text{psum}[i] = \text{psum}[i-1] + A[i]}$$

arr →

3		9		4		-5		2
0		1		2		3		4

psum =

3		12		16		11		13
0		1		2		3		4



int[] prefixSum(Pnt[] arr) {

    ↙ Pnt 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]

        3

n=4

    return psum;

3

arr:

3		9		4		-5
0		1		2		3

psum: ↗

3		12		16		11
0		1		2		3

out of  
loop

↑  
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 SC.

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;
}

```

n: 4 byte  
i: 4 byte  
total space: 8 byte  
 $S \cdot C = O(1)$

## - Range Sum Query

**Problem 2. Range Sum queries**

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

Constraints :  
 $1 \leq n \leq 10^5$       in single query,  $L-R$   
 $1 \leq Q \leq 10^5$       find sum from  $L\text{-index}$  to  $R\text{-index}$ .

$A = \begin{bmatrix} 3 & 4 & -2 & 6 & 8 & 10 & 13 & 1 \end{bmatrix}$        $0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7$

$Q = 4$  ( $L \leq R$ )

left	Right
1	3
2	6
5	5
0	2

query:

I  $[1] \leftrightarrow [3] \rightarrow 8$       left for 1<sup>st</sup> query  $\Rightarrow \text{query}[i][0]$   
II  $[2] \leftrightarrow [6] \rightarrow 35$       right for 1<sup>st</sup> query  $\Rightarrow \text{query}[i][1]$   
III  $[5] \leftrightarrow [5] \rightarrow 10$   
IV  $[0] \leftrightarrow [3] \rightarrow 11$       left for 4<sup>th</sup> query  $\Rightarrow \text{query}[i][0]$   
right for 4<sup>th</sup> query  $\Rightarrow \text{query}[i][1]$

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 7<sup>th</sup> over = 65

Score in end of 6<sup>th</sup> over = 49

Runs scored in 7<sup>th</sup> over =  $65 - 49 = 16$   $\approx$

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]

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

sum scored in ending of 10<sup>th</sup> over = 97

sum scored in ending of 5<sup>th</sup> over = 31

$$\text{difference} = 97 - 31$$

$$= 66 \quad \underline{\underline{\text{Ans}}}$$

~~~~~ Problem 2. Range Sum queries ~~~~

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

Constraints :

$$1 \leq n \leq 10^5 \rightarrow 10^5$$

$$1 \leq Q \leq 10^5 \rightarrow 10^5$$

$$O(n+q) = 10^5 + 10^5 = 2 \cdot 10^5 \rightarrow \text{perf. Appr.}$$

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

$$Q = 4 \ (L \leq R)$$

Brute force Approach:

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

- ① Every possible query,  
Extract L & R from query.
- ② find sum from L-index to min(R)

```
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 by
    // & fill it in ans[i]
    int sum = 0;
    for (int j = l; j <= r; j++)
        sum += A[j];
    ans[i] = sum;
}

```

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

Improvement:

Local Sym(0,6) 111111111111

12 / Sym(0,2) / 13

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

$$psum[6] = sum_{in A}(0 to 6)$$

psumm →

|   |   |   |    |    |    |    |    |
|---|---|---|----|----|----|----|----|
| 2 | 7 | 5 | 11 | 19 | 29 | 42 | 43 |
| 0 | 1 | 2 | 3  | 4  | 5  | 6  | 7  |

$$\text{sum}(3,6) = \text{sum}(0,6) - \text{sum}(0,2)$$

$$= \text{psvm}[6] - \text{psvm}[2]$$

$$\text{psum}[6] = A[0] + A[1] + A[2] + A[3] + A[4] + A[5] + A[6]$$

$$-\text{psum}[2] = A[0] + A[1] + A[2]$$

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

sum from 3 to 6

TODO:

$$\text{sum}(2, 5) = \text{sum}(0, 5) - \text{sum}(0, 1)$$

$$= \text{psum}[5] - \text{psum}[1]$$

$$\begin{aligned} \text{psum}[5] &= A[0] + A[1] + A[2] + A[3] + A[4] + A[5] \\ \text{psum}[1] &= A[0] + A[1] \\ \hline &= A[2] + A[3] + A[4] + A[5] \\ \text{left} \\ \downarrow \\ \text{sum}(0, 3) &= \text{psum}[3] - \text{psum}[1] \quad \times \\ \hookrightarrow \text{psum}[3] \end{aligned}$$

$$\boxed{\text{sum}(L, R) = \text{psum}[R] - \text{psum}[L-1] \quad \text{where } L \neq 0}$$

$\text{sum}(L, R)$ :

```

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

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];      Space comp. SC: O(n)
        } else {
            ans[i] = psum[r] - psum[l-1];
        }
    }

    return ans;
}
    
```

Book 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 :

$$\text{Sum of all the elements in left side of } i = \text{Sum of all the elements in right side of } i \quad \} i \text{ is equilibrium index}$$

Ex1

|       |                                                             |
|-------|-------------------------------------------------------------|
| $A =$ | $\begin{bmatrix} -7 & 5 & 1 & 2 & -4 & 3 & 0 \end{bmatrix}$ |
|       | $\begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 \end{matrix}$     |

|       |                                                   |
|-------|---------------------------------------------------|
| $A =$ | $\begin{bmatrix} 5 & 1 & 3 & 2 & 9 \end{bmatrix}$ |
|       | $\begin{matrix} 0 & 1 & 2 & 3 & 4 \end{matrix}$   |

|       |                                           |
|-------|-------------------------------------------|
| $A =$ | $\begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$ |
|       | $\begin{matrix} 0 & 1 & 2 \end{matrix}$   |

$$\begin{aligned} lsum &= 0 \\ rsum &= 7 + 2 + 1 \\ \text{equilibrium index} &= 2 \end{aligned}$$

$$\begin{aligned} lsum &= 0 \\ rsum &= 5 + 1 + 9 \\ \text{equilibrium index} &= 3 \end{aligned}$$

$$\begin{aligned} lsum &= 0 \\ rsum &= 2 + 0 \end{aligned}$$

if ( $lsum \neq rsum$ ) End  $\rightarrow$  equilibrium index  $\Rightarrow -1$ .

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 iterate on every index of array.
- ② for every possible array calculate left sum & right sum
- ③ if  $lsum \neq rsum$  is equal return index.
- ④ return -1.

TODD:  i

|                                                                | Itr by<br>① | Br by<br>② | Total Str. |
|----------------------------------------------------------------|-------------|------------|------------|
| <code>for(int i=0; i&lt;n; i++) {</code>                       |             |            |            |
| // calculate left sum                                          |             |            |            |
| <code>    for(int j=0; j&lt;i; j++) { lsum += A[j]; }</code>   |             |            |            |
| // calculate right sum:                                        |             |            |            |
| <code>    for(int j=i+1; j&lt;n; j++) { rsum += A[j]; }</code> |             |            |            |
| if(lsum == rsum)                                               |             |            |            |
| <code>        return i;</code>                                 |             |            |            |
| }                                                              |             |            |            |
| <code>return -1;</code>                                        |             |            |            |

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

### Optimisation:

Sum of all the elements which are left of  $i$  = sum of all element in right side of  $i$

$\text{psum}[i-1]$

$\text{psum}[n-1] - \text{psum}[i]$

`int equilibriumIndex(int[] A) {`

`int[] psum = prefixSum(A);` n ite.  
`int n = A.length;` n space

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

`int lsum = 0;  
        if(i > 0) {  
            lsum = psum[i-1];  
        }  
        int rsum = psum[n-1] - psum[i];  
        if(lsum == rsum) {  
            return i;  
        }`

3  
3  
3  
3  
return -1;

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

|   |   |   |   |   |
|---|---|---|---|---|
| 5 | 4 | 3 | 2 | 9 |
| 0 | 1 | 2 | 3 | 4 |

|   |   |   |    |    |
|---|---|---|----|----|
| 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[]

$$\text{sum}(2, 4) \Rightarrow p\text{sum}[4] - p\text{sum}[1]$$

$$p\text{sum}[R] - p\text{sum}[L-1]$$

$$p\text{sum}[N-1] - p\text{sum}[i+1-x] \Rightarrow p\text{sum}[N-1] - p\text{sum}[0]$$

Quiz 5:  $ar[7] = \{-7, 1, 5, 2, -4, 3, 0\}$

How many equilibrium indices are there?

|         | -7 | 1  | 5  | 2  | -4 | 3  | 0 |  |
|---------|----|----|----|----|----|----|---|--|
| Indices | 0  | 1  | 2  | 3  | 4  | 5  | 6 |  |
| psum    | -7 | -6 | -1 | 1  | -3 | 0  | ① |  |
| lsum    | 0  | -7 | -6 | -1 | 1  | -2 | 0 |  |
| rsum    | 7  | 6  | 1  | -1 | 2  | 0  | 0 |  |

$$lsum = 0$$

$$if(i > 0)$$

$$lsum = psum[i-1]$$

3

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

↑

$$0 - (-7) = 7$$

$$\text{count} = \cancel{0} + 2$$

2, 6  $\rightarrow$  one equilibrium  
indice

$$\text{count} = 2$$

$$psum[n-1] - psum[0] \\ = 0 - 0 = 0$$

$$0 - (-7) = 7$$

Basic property

$$a - (-b) = a + b$$

$$a - b = a - b$$

$$a - (+b) = a - b$$

$$-a - b = -(a + b)$$

$$-2 - 3 = -(2 + 3) \\ = \cancel{(-5)}$$

## - Even Number for Q query

Problem 4 : Even Numbers for Q query.

Given an array  $A[]$  of size  $N$ , and  $Q$  different queries. Find the count of even numbers for every  $Q$  query.

|       |   |   |   |   |    |    |    |    |
|-------|---|---|---|---|----|----|----|----|
| $A =$ | 3 | 5 | 8 | 9 | 16 | 14 | 13 | 12 |
|       | 0 | 1 | 2 | 3 | 4  | 5  | 6  | 7  |

Queries :

- |     |     |            |     |
|-----|-----|------------|-----|
| $L$ | $R$ | <u>ans</u> | $q$ |
| 1   | 5   | 2          | 0   |
| 2   | 6   | 3          | 1   |
| 4   | 5   | 2          | 2   |
| 4   | 4   | 1          | 3   |
| 3   | 6   | 2          | 4   |

|   |   |
|---|---|
| 0 | 1 |
| 1 | 5 |
| 2 | 6 |
| 3 | 5 |
| 4 | 4 |
| 5 | 6 |

To Do:

# How to solve  
this problem in  
 $O(n * q)$  complexity

$$\text{left} = \text{query}[i][0]$$

$$\text{right} = \text{query}[i][1]$$

|      |   |   |   |   |    |    |    |    |
|------|---|---|---|---|----|----|----|----|
| Arr: | 3 | 5 | 8 | 9 | 16 | 14 | 13 | 12 |
|      | 0 | 1 | 2 | 3 | 4  | 5  | 6  | 7  |

prefixCount



Even :

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 2 | 3 | 3 | 4 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

prefixCount[i]  $\Rightarrow$  count of Even number from 0 to i index

Query       $2, 5 \rightarrow \text{prefixCount}[5] - \text{prefixCount}[1]$   
 count from 0 to 5      count from 0 to 1

$$\text{count}(L, R) = \text{prefixCount}[R] - \text{prefixCount}[L-1] \quad L \neq 0$$

Future  
To Do:

Hint       $A[0] = 3, \text{pcount}[0] = 0, A[0] = 4, \text{pcount}[0] = 1$

**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 means 2D array of  $2 \times 2$  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
- Subarray from s to e
- Print All subarray
- Sum of every subarray
- sum of all subarray's sum

### Introduction of subarray

Subarray: continuous part of an array is known as subarray.

|   |   |   |   |   |   |   |   |    |  |   |  |   |  |   |  |    |
|---|---|---|---|---|---|---|---|----|--|---|--|---|--|---|--|----|
| 2 |   | 3 |   | 9 |   | 4 |   | -5 |  | 6 |  | 7 |  | 9 |  | 10 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8  |  |   |  |   |  |   |  |    |

[9, 4, -5, 6, 7] → valid

[9, 4, 6, 7, 9] → invalid subarray

1. A single element from an array can be subarray.

2. Complete array also can be a subarray.

3. Order of Elements matter a lot

## Subarray from s to e

|         |                                                                                                                                                                                                                                         |   |   |    |   |    |   |    |   |    |   |   |   |   |   |   |   |   |   |
|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---|---|----|---|----|---|----|---|----|---|---|---|---|---|---|---|---|---|
| array → | <table border="1"> <tr> <td>2</td><td>3</td><td>9</td><td>4</td><td>-5</td><td>6</td><td>7</td><td>9</td><td>10</td></tr> <tr> <td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td></tr> </table> | 2 | 3 | 9  | 4 | -5 | 6 | 7  | 9 | 10 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 2       | 3                                                                                                                                                                                                                                       | 9 | 4 | -5 | 6 | 7  | 9 | 10 |   |    |   |   |   |   |   |   |   |   |   |
| 0       | 1                                                                                                                                                                                                                                       | 2 | 3 | 4  | 5 | 6  | 7 | 8  |   |    |   |   |   |   |   |   |   |   |   |

A particular subarray can be defined by some starting index 's' and some ending index 'e'.

$$s=2 \quad \text{subarray: } [9, 4, -5, 6, 7]$$

$$e=6$$

length of subarray with  $s \leq e$

$$\boxed{\text{length of Subarray} = e - s + 1}$$

$$\begin{aligned} \text{for example} &= 6 - 2 + 1 \\ &= 4 + 1 = 5 \end{aligned}$$

|       |                                                                                                                                   |   |   |   |   |   |   |   |   |
|-------|-----------------------------------------------------------------------------------------------------------------------------------|---|---|---|---|---|---|---|---|
| array | <table border="1"> <tr> <td>3</td><td>1</td><td>4</td><td>5</td></tr> <tr> <td>0</td><td>1</td><td>2</td><td>3</td></tr> </table> | 3 | 1 | 4 | 5 | 0 | 1 | 2 | 3 |
| 3     | 1                                                                                                                                 | 4 | 5 |   |   |   |   |   |   |
| 0     | 1                                                                                                                                 | 2 | 3 |   |   |   |   |   |   |

| s | e | subarray |
|---|---|----------|
| 0 | 0 | 3        |
| 0 | 1 | 3 1      |
| 0 | 2 | 3 1 4    |
| 0 | 3 | 3 1 4 5  |

| s | e | subarray |
|---|---|----------|
| 1 | 1 | 1        |
| 1 | 2 | 1 4      |
| 1 | 3 | 1 4 5    |

| s | e | subarray |
|---|---|----------|
| 2 | 2 | 4        |
| 2 | 3 | 4 5      |

| s | e | subarray |
|---|---|----------|
| 2 | 3 | 5        |

### Problem 1 : SubArray from s to e

Given an array arr[], print subarray starting from index s and ending at index e.

Example

$$S=1$$

|         |                                                                                                                                                                           |   |   |   |   |   |   |   |   |   |   |   |   |
|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---|---|---|---|---|---|---|---|---|---|---|---|
| array → | <table border="1"> <tr> <td>3</td><td>4</td><td>5</td><td>9</td><td>7</td><td>6</td></tr> <tr> <td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr> </table> | 3 | 4 | 5 | 9 | 7 | 6 | 0 | 1 | 2 | 3 | 4 | 5 |
| 3       | 4                                                                                                                                                                         | 5 | 9 | 7 | 6 |   |   |   |   |   |   |   |   |
| 0       | 1                                                                                                                                                                         | 2 | 3 | 4 | 5 |   |   |   |   |   |   |   |   |

$$e=4$$

↑ → loop step

Steps: 1. Start loop from s to e and print every element.

```
void printsubarray (int A[], int s, int e) {
```

o/p screen:

```
    for (int i = s; i <= e; i++) {
        cout << A[i] << " ";
    }
```

4 5 9 7

}

## Print All subarray

arr : 

|    |    |    |
|----|----|----|
| 10 | 20 | 30 |
| 0  | 1  | 2  |

$$n=3, \text{ count of Subarrays} = \frac{n(n+1)}{2} = \frac{3(3+1)}{2} = 6$$

count of subarray which are starting from 0<sup>th</sup> index:

|    |    |    |   |          |
|----|----|----|---|----------|
| 10 |    |    | } | count: 3 |
| 10 | 20 |    |   |          |
| 10 | 20 | 30 |   |          |

count of subarray which are starting from 1<sup>st</sup> index:

|    |    |   |          |
|----|----|---|----------|
| 20 |    | } | count: 2 |
| 20 | 30 |   |          |

count of subarray which are starting from 2<sup>nd</sup> index:

|    |  |   |          |
|----|--|---|----------|
| 30 |  | } | count: 1 |
|----|--|---|----------|

$$\text{Total Count} = 3 + 2 + 1 = 6 ]$$

(length of array = 3)

Given arr array with length n :

A: [ a<sub>0</sub>, a<sub>1</sub>, a<sub>2</sub>, a<sub>3</sub>, ..., . . . , a<sub>n-1</sub> ]

no. of subarrays starting from 0<sup>th</sup> index = n

no. of subarrays starting from 1<sup>st</sup> index = n-1

no. of subarrays starting from 2<sup>nd</sup> index = n-2

⋮

(n-2)<sup>th</sup> index = 2

(n-1)<sup>th</sup> index = 1

$$\text{Total Count of subarray} = 1 + 2 + 3 + 4 + \dots + n-2 + n-1 + n$$

$$\text{Sum of } n \text{ natural number} = \frac{n(n+1)}{2}$$

### Problem 2 : Print All subarrays

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

|   |    |   |
|---|----|---|
| 3 | -1 | 2 |
| 0 | 1  | 2 |

$$\begin{aligned} \text{total subarray} &= \frac{n(n+1)}{2} \\ &= \frac{3 \times 4}{2} = 6 \end{aligned}$$

S= 0

(0, 0) - 3

(0, 1) - 3 -1

(0, 2) - 3 -1 2

i = 0  
to  
n-1

S= 1

(1, 0) - -1

(1, 1) - -1 2

j = i to n-1  
 $\left\{ \begin{array}{l} S=0 \longrightarrow e \in 0, 1, 2 \\ S=1 \longrightarrow e \in 1, 2 \\ S=2 \longrightarrow e \in 2 \end{array} \right.$

S= 2

(2, 0) - 2

r

```
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++) {
```

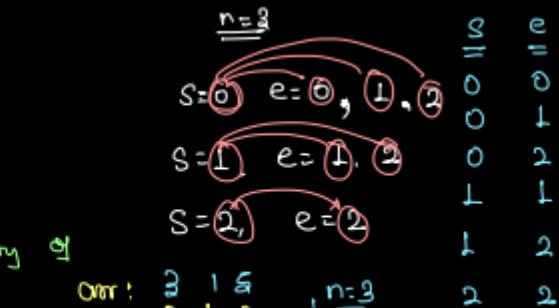
```
            System.out.print(A[i] + " ");
```

```
        }
```

```
        System.out.println();
```

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

S.C:  $O(1)$



|   |   | n=3               | S | e |
|---|---|-------------------|---|---|
| S | e | arr: 3 -1 2 , n=3 | 0 | 0 |
| 0 | 0 | (0,0) → 3         | 0 | 0 |
| 0 | 1 | (0,1) → 3 -1      | 0 | 1 |
| 0 | 2 | (0,2) → 3 -1 2    | 0 | 2 |
| 1 | 1 | (1,1) → -1        | 1 | 1 |
| 1 | 2 | (1,2) → -1 2      | 1 | 2 |
| 2 | 2 | (2,2) → 2         | 2 | 2 |

## Sum of every subarray

### Problem 3 : Sum of every subarray

Given an array arr[], print sum of every possible subarray

|       |                                                                                                                                                                       |   |    |   |   |   |   |
|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|---|----|---|---|---|---|
| arr : | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>3</td><td>-1</td><td>4</td></tr> <tr> <td>0</td><td>1</td><td>2</td></tr> </table> | 3 | -1 | 4 | 0 | 1 | 2 |
| 3     | -1                                                                                                                                                                    | 4 |    |   |   |   |   |
| 0     | 1                                                                                                                                                                     | 2 |    |   |   |   |   |

| s | e | subarray   | sum | o/p |
|---|---|------------|-----|-----|
| 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   |     |

```
void printSumOfEverySubarray ( int arr[] ) {
```

```
    int n = arr.length;
```

```
    for (int s=0; s<n; s++) {
```

```
        for (int e=s; e<n; e++) {
```

```
            // find sum b/w s to e
```

```
            int sum=0;
```

```
            for (int i=s; i<=e; i++) {
```

```
                sum+=arr[i];
```

```
}
```

```
            System.out.println(sum);
```

```
    }
```

T.C : O(n<sup>3</sup>)

S.C : O(1)

## Improvisation

Allowed T.C : O(n<sup>2</sup>)

Improvisation:

$$\begin{aligned}
 & \xrightarrow{s=0} \text{sum b/w } (s,e) = psum[e] \\
 (s,e) \\
 & \xrightarrow{s_1=0} \text{sum b/w } (s_1,e) = psum[e] - psum[s_1]
 \end{aligned}$$

technique: prefix sum

```

void printSumOfEverySubarray (int arr[])
{
    int n = arr.length;
    int[] psum = prefixSum(arr); [O(n)]
    for (int s=0; s<n; s++) {
        for (int e=s; e<n; e++) {
            // find sum b/w s to e
            if (s==0) {
                sOpIn(psum[e]);
            } else {
                sOpIn (psum[e] - psum[s-1]);
            }
        }
    }
}

```

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

s  
psum      s to e

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

$$S.C. = O(n)$$

|   |   |    |   |
|---|---|----|---|
| 2 | 2 | -4 | 6 |
| 0 | 1 | 2  | 3 |

|   |   |   |   |
|---|---|---|---|
| 2 | 5 | 1 | 7 |
| 0 | 1 | 2 | 3 |

n=4

| S | e | sum                         | p[e] |
|---|---|-----------------------------|------|
| 0 | 0 | $p[0]$                      | ③    |
| 0 | 1 | $p[0] + p[1]$               | ④    |
| 0 | 2 | $p[0] + p[1] + p[2]$        | ①    |
| 0 | 3 | $p[0] + p[1] + p[2] + p[3]$ | ⑦    |
| 1 | 1 | $p[1]$                      | ②    |
| 1 | 2 | $p[1] + p[2]$               | ⑥    |
| 1 | 3 | $p[1] + p[2] + p[3]$        | ④    |
| 2 | 2 | $p[2]$                      | ⑤    |
| 2 | 3 | $p[2] + p[3]$               | ②    |
| 3 | 3 | $p[3]$                      | ⑥    |

Can we reduce space complexity??

further improvisation: carry forward technique

array: 

|   |   |    |   |   |
|---|---|----|---|---|
| 3 | 4 | -2 | 5 | 1 |
| 0 | 1 | 2  | 3 | 4 |

$s=1, e=1, 2, 3, 4$

$(s, e)$       sum

(1, 1) sum =  $\underbrace{A[1]}_{\text{sum}} \Rightarrow s$

(1, 2) sum =  $\underbrace{A[1]}_{s} + \underbrace{A[2]}_{\text{sum}} \Rightarrow s$

(1, 3) sum =  $\underbrace{A[1]}_{s} + \underbrace{A[2]}_{\text{sum}} + \underbrace{A[3]}_{\text{sum}} \Rightarrow s$

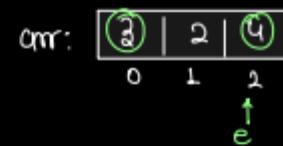
(1, 4) sum =  $\underbrace{A[1]}_{s} + \underbrace{A[2]}_{\text{sum}} + \underbrace{A[3]}_{\text{sum}} + \underbrace{A[4]}_{\text{sum}} \Rightarrow s$

```

void solve(int arr[]) {
    int n = arr.length;
    for(int s=0; s < n; s++) {
        int sum=0;
        for(int e=s; e < n; e++) {
            sum += A[e];
            System.out.println(sum);
        }
    }
}
T.C: O(n^2)
S.C: O(1)

```

Most simplified version:



| s | e | sum | o/p sum |
|---|---|-----|---------|
| 0 | 0 | 3   | 3 ✓     |
|   | 1 | 5   | 5 ✓     |
| 1 | 2 | 9   | 9 ✓     |
|   | 2 | 2   | 2 ✓     |
| 2 | 2 | 6   | 6 ✓     |
|   | 2 | 4   | 4 ✓     |

$$O(1) \left\{ \begin{array}{l} l \text{ to } 10 = 10 - 1 + 1 \\ = 10 \\ a \text{ to } b = b - a + 1 \end{array} \right.$$

possible numbers from 1 to  $\infty$   
 $\rightarrow 1 \text{ to } 10$   
 $O(n)$ .

## 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

$$\text{sum} = x * A[0] + y * A[1] + Z * A[2]$$

### Problem 4 : Sum of all subarrays sum.

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

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

| s | e | Subarray   | sum |                               |
|---|---|------------|-----|-------------------------------|
| 0 | 0 | [3]        | 3   | 0[p]                          |
| 0 | 1 | [3, -1]    | 2   |                               |
| 0 | 2 | [3, -1, 4] | 6   | Sum: 3 + 2 + 6 + (-1) + 2 + 4 |
| 1 | 1 | [-1]       | -1  |                               |
| 1 | 2 | [-1, 4]    | 3   | = 17 Ans                      |
| 2 | 2 | [4]        | 4   |                               |

brute force approach:

```

int solve(int A[], int n) {
    int ans = 0;
    int n = A.length;
    for(int s=0; s<n; s++) {
        int sum = 0;
        for(int e=s; e<n; e++) {
            sum += A[e];
            System.out.println(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   | $\rightarrow A[0]$               |
| 0 | 1 | [3, -1]    | 2   | $\rightarrow A[0] + A[1]$        |
| 0 | 2 | [3, -1, 4] | 6   | $\rightarrow A[0] + A[1] + A[2]$ |
| 1 | 1 | [-1]       | -1  | $A[1]$                           |
| 1 | 2 | [-1, 4]    | 3   | $A[1] + A[2]$                    |
| 2 | 2 | [4]        | 4   | $A[2]$                           |

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

$$3 * 3 + 4 * (-1) + 2 * 4$$

$$= 9 - 4 + 8 = 13 \quad \text{Ans}$$

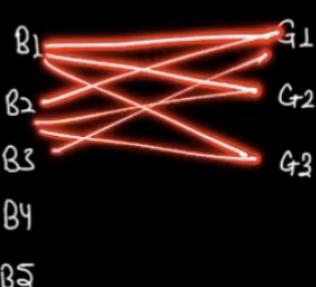
$$\text{Sum} = x * A[0] + y * A[1] + z * A[2]$$

How many time  $i^{\text{th}}$  element is coming in overall sum.



valid Ending point  $\Rightarrow e = i, i+1, i+2, \dots, n-1$

Group Bays



Group Grids

Different possibility  
for pair up:

Starting pt

$i+1$

Ending:

$n-i$

|                                                      |                   |
|------------------------------------------------------|-------------------|
| count of<br>{th index in overall<br>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;  
}
```

$n=3$

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

|   |    |   |
|---|----|---|
| 2 | -1 | 4 |
| 0 | 1  | 2 |

| i | $arr[i] * (i+1)(n-i)$   |
|---|-------------------------|
| 0 | $3 * (1) * (3) = 9$     |
| 1 | $(-1) * (2) * (2) = -4$ |
| 2 | $4 * (3) * (1) = 12$    |

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;  
}
```

$n=3$

|   |    |   |
|---|----|---|
| 2 | -1 | 4 |
| 0 | 1  | 2 |

| i | $arr[i] * (i+1)(n-i)$   |
|---|-------------------------|
| 0 | $3 * (1) * (3) = 9$     |
| 1 | $(-1) * (2) * (2) = -4$ |
| 2 | $4 * (3) * (1) = 12$    |

ans = 17 ↗

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

Imp Observation of prefix sum -----

Continuous Sum Query -----

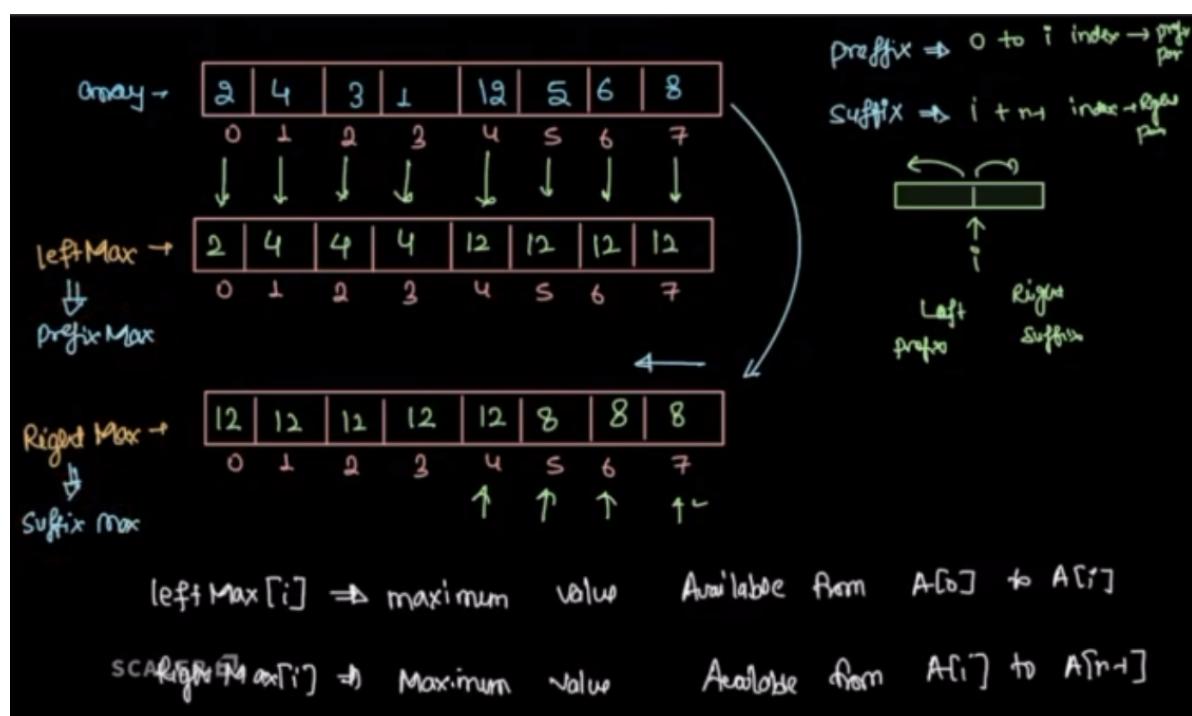
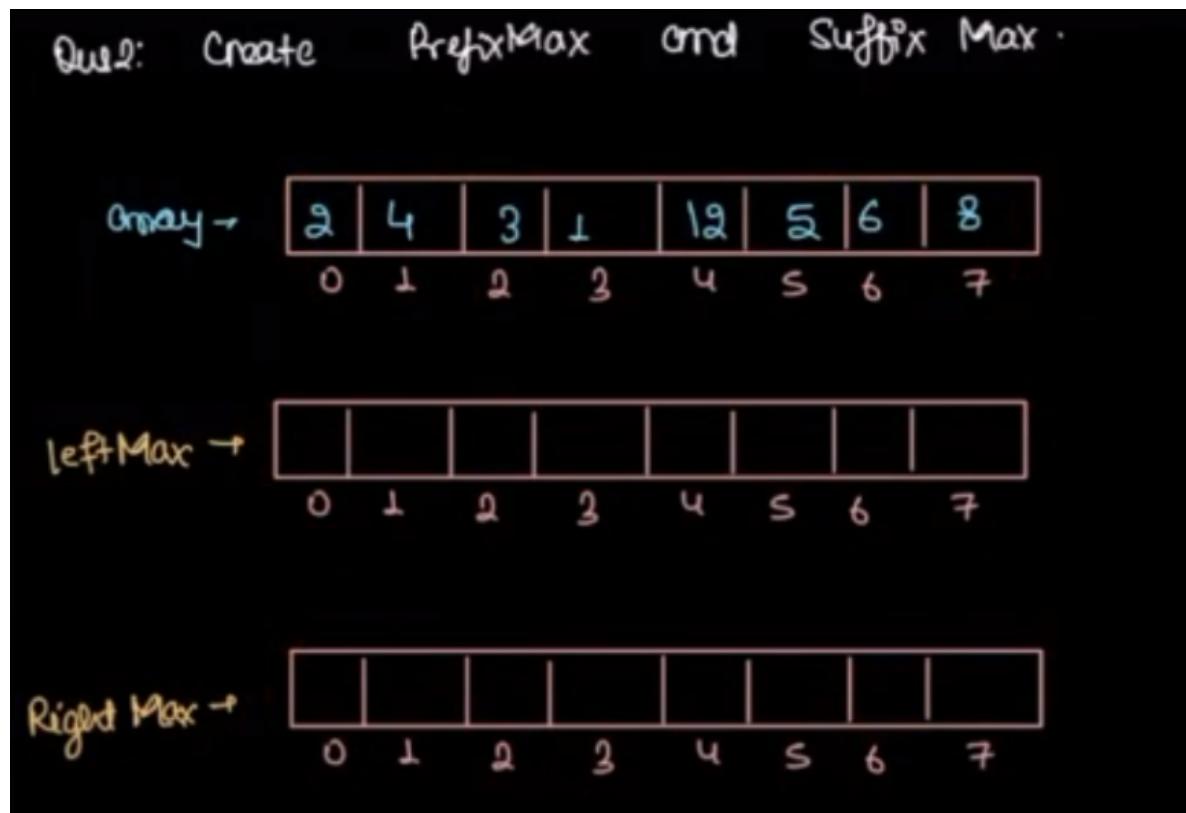
Leftmax and Rightmax -----

RainWater Trapping -----

Kadane's Algorithm -----

Leftmax and Rightmax -----

Create PrefixMax (leftMax) and Suffix Max (rightMax)



## Left Max

```

int[] leftMax(int[] A) {
    int n = A.length;
    int lmax = new int[n];
    lmax[0] = A[0];
    for(int i=1; i<n; i++) {
        lmax[i] = Math.max(lmax[i-1], A[i]);
    }
    return lmax;
}

```

⑩   7   12   4  
↓   2   3  
0   1   2   3
   
 $n=4 \quad l_{\max} = \boxed{10 \mid 10 \mid 12 \mid 12}$   
 $\uparrow$   
i  
previous max   current element

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

```

int[] rightMax(int[] A) {
    int n = A.length;
    int rmax = new int[n];
    rmax[n-1] = A[n-1];
    for(int i=n-2; i>=0; i--) {
        rmax[i] = Math.max(rmax[i+1], A[i]);
    }
    return rmax;
}

```

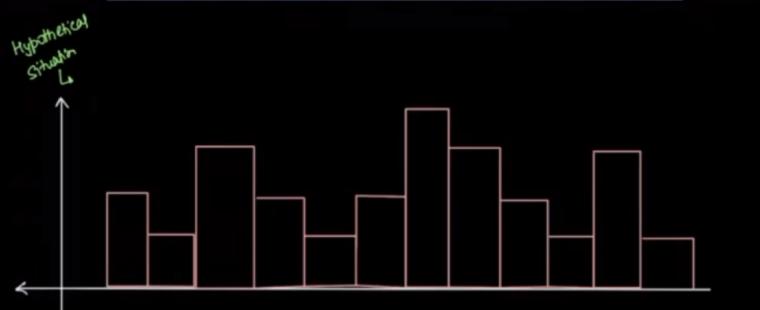
13   7   12   2   9  
↓   2   3   4  
0   1   2   3   4
   
 $n=5 \quad r_{\max} = \boxed{13 \mid 12 \mid 12 \mid 9 \mid 9}$   
 $\uparrow$   
i  
max till now in next   current element

## RainWater Trapping -----

Ques 3. Rainwater Trapping  
 Given an array arr[], where arr[i] denotes height of ith building.  
 Return amount of water trapped on all buildings.

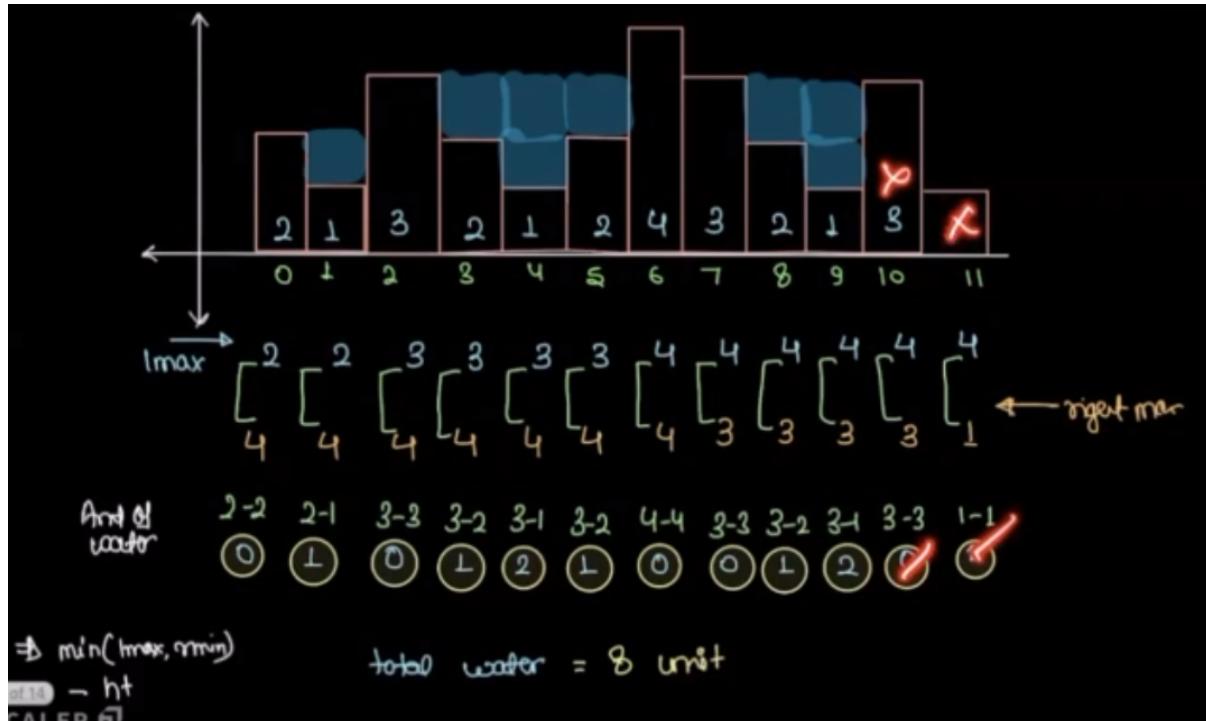
arr[] : 

|   |   |   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 1 | 3 | 2 | 1 | 2 | 4 | 3 | 2 | 1 | 3 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|



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

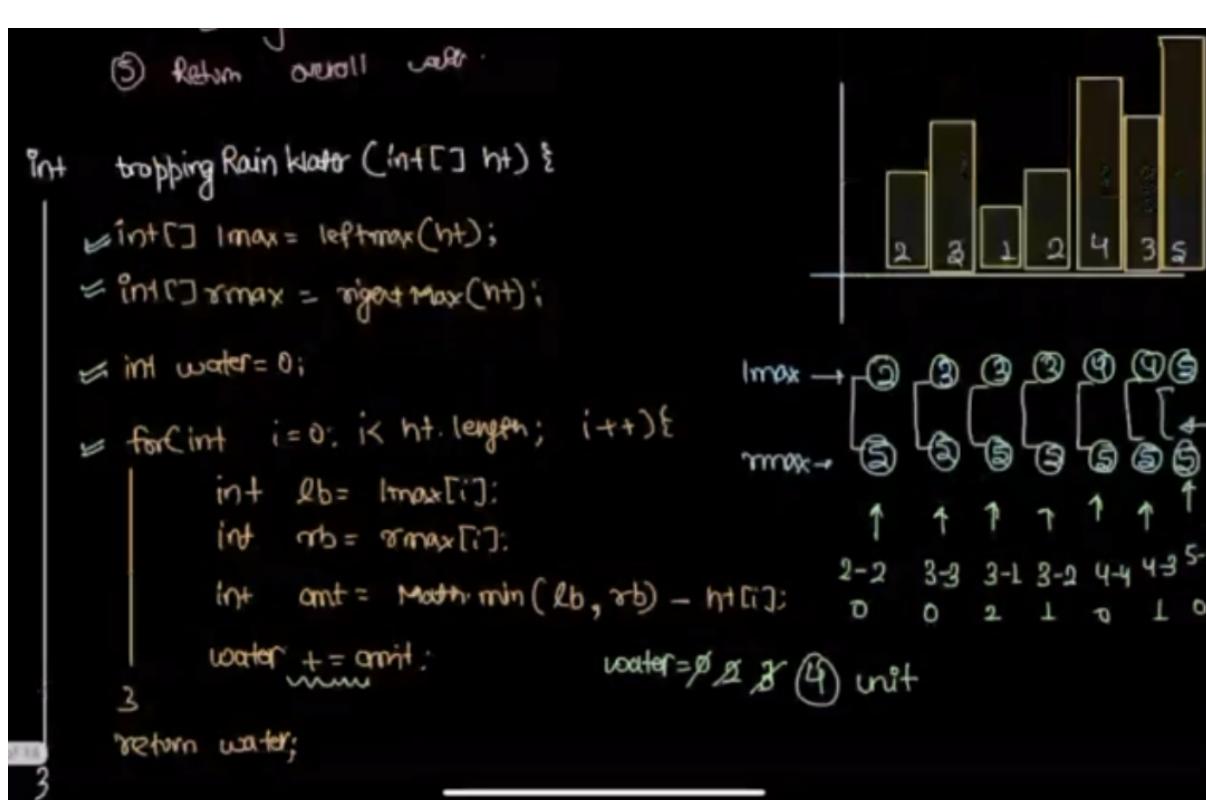
Note -> Want to calculate amount of water of ith building take help of leftMax[i] and rightMax[i]



## Steps

1. Calculate leftMax
2. Calculate rightMax
3. Iterate 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 columnwise sorted matrix

### Print max subarray of len=k

Q1. Given  $ar[N]$  elements, print max subarray sum of len = k

Constraints

$$1 \leq N \leq 10^5$$

$$1 \leq K \leq N$$

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

$$ar[10] : \begin{array}{cccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ -3 & 4 & -2 & 5 & 3 & -2 & 8 & 2 & -1 & 4 \end{array} \quad K=5$$

s    e    sum

0    4    7

1    5    8

2    6    12

Ans = 16

3    7    16

4    8    10

5    9    11

Idea 1 → For every subarray of length k,  
iterate & calculate the sum,  
& update the overall maximum

$TC = \text{No. of subarrays of } * \text{ TC for each length } k$

$$TC = \left( (N - k+1) * k \right)$$

The diagram illustrates the range of the variable  $k$  in the summation. It shows two horizontal arrows pointing downwards from  $k=1$  to  $k=N/2$  and from  $k=N$  back to  $k=1$ , indicating the bounds of the loop iteration.

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

$$= O(N) \quad = \left(\frac{N}{2}+1\right)*\frac{N}{2}$$

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

$$TC \leq O(N^2)$$

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

```

int maxsubarraysum ( int [] ar, int k)
    // Create psum array | { TODO } TC:O(N)
    s=0 , e=k-1
    long ans = Integer.MIN_VALUE;
    while ( e < n )
        long sum=0 // subarr [s e]
        if ( s == 0 ){
            sum = psum[e]
        } else {
            sum = psum[e] - psum[s-1];
        }
        if ( sum > ans ) { ans = sum };
        s = s+1
        e = e+1
    }
    return ans
    SC: O(N)
    TC: O(N)

```

3

$$TC = N + N - k + 1 = 2N - \underline{k+1}$$

$$\approx O(N)$$

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

|   |   |                                                   | 0 1 2 3 4 5 6 7 8 9    |
|---|---|---------------------------------------------------|------------------------|
|   |   |                                                   | 3 4 -2 5 3 -2 8 2 -1 4 |
|   |   |                                                   | [0-5] sum=11           |
| s | e | sum                                               | sum=16                 |
| 0 | 5 | 11                                                | sum=14                 |
| 1 | 6 | sum = sum - ar[0] + ar[6] = 11 - 3 + 8 = 16       |                        |
| 2 | 7 | sum = sum - ar[1] + ar[7] = 16 - 4 + 2 = 14       |                        |
| 3 | 8 | sum = sum - ar[2] + ar[8] = 14 - (-2) + (-1) = 15 |                        |
| 4 | 9 | sum = sum - ar[3] + ar[9] = 15 - 5 + 4 = 14       |                        |
|   |   |                                                   | Ans = 16               |

```

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

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

    if ( sum > ans ) ans = sum;

    s=1 e=k

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

    return ans;
}

```

$T.C = K + n - K = O(n)$   
 $S.C = O(1)$

Note: Fixed size subarray  $\rightarrow$  Sliding window - print boundary in clockwise direction

Q2 Given  $mat[n][n]$ , print boundary in clockwise direction

Constraints

$$1 \leq N \times N \leq 10^6$$

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

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 |

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

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

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 =  $\rightarrow n=5$

01. Print 4 ele in 0<sup>th</sup> row = L→R
02. Print 4 ele in 4<sup>th</sup> col = T→D
03. Print 4 ele in 4<sup>th</sup> row = R→L
04. Print 4 ele in 0<sup>th</sup> 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 0<sup>th</sup> row L→R
02. Print 2 ele in 2<sup>nd</sup> col T→D
03. Print 2 ele in 2<sup>nd</sup> row R→L
04. Print 2 ele in 0<sup>th</sup> col D→T

void printboundary (int arr[ ][ ] 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
}

```

| K | i | j | <u><math>\frac{k \times n}{1 \leq k \leq n}</math></u> |
|---|---|---|--------------------------------------------------------|
| 1 | 0 | 0 | $2 \leq 5$                                             |
| 2 | 0 | 1 | $3 \leq 5$                                             |
| 3 | 0 | 2 | $4 \leq 5$                                             |
| 4 | 0 | 3 | $5 \leq 5$                                             |
| 5 | 0 | 4 | $5 \leq 5$                                             |

Constraints  
 $1 \leq n \leq 10^6$   
 $-10^6 \leq ar[i] \leq 10^6$

```

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

```

TC: O(n)

SC: O(1)

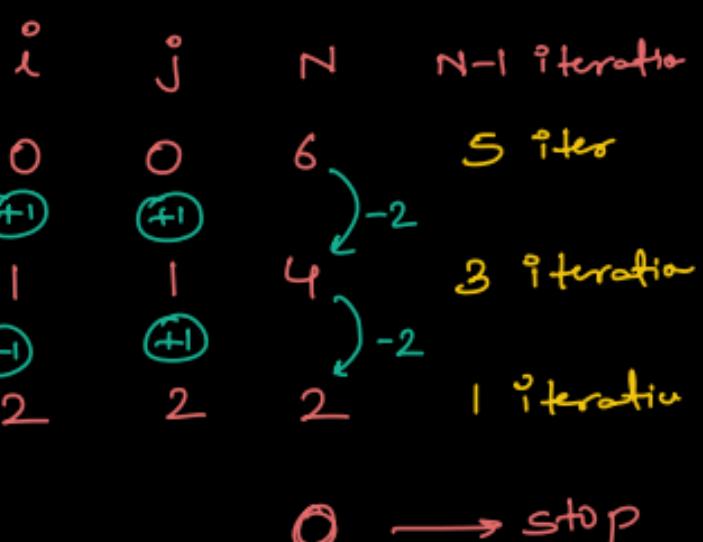
if (n==1) { print (ar[i][j]) } → Edge case

3

## spiral printing

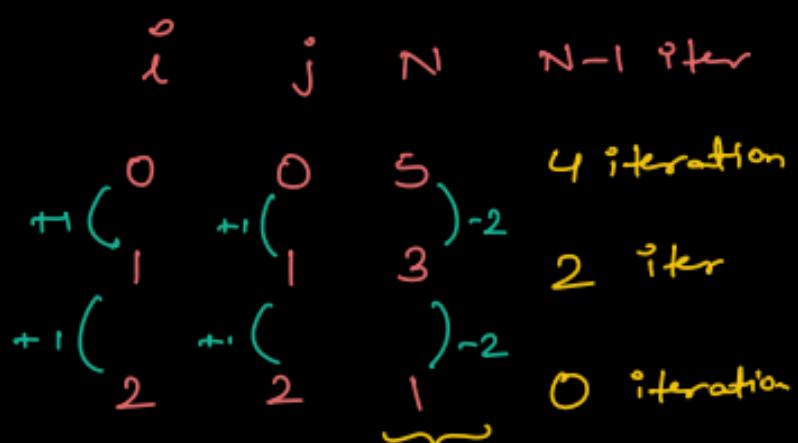
$\text{mat}[6][6] \curvearrowleft N = 6$

|   | 0  | 1  | 2  | 3  | 4  | 5  |
|---|----|----|----|----|----|----|
| 0 | 1  | 2  | 3  | 4  | 5  | 6  |
| 1 | 20 | 21 | 22 | 23 | 24 | 7  |
| 2 | 19 | 32 | 33 | 34 | 25 | 8  |
| 3 | 18 | 21 | 36 | 35 | 26 | 9  |
| 4 | 17 | 30 | 29 | 28 | 27 | 10 |
| 5 | 16 | 15 | 14 | 13 | 12 | 11 |



$\text{mat}[5][5] \curvearrowleft N = 5$

|   | 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 |



↳ Edge case

Testcase

$3 * 4$



$i \ j \ n \ n-1$

```
void spiral printing (int arr[N][N])
```

```
    int i=0, j=0
```

```
    while (n>1)
```

```
        for (k=1; k<n; k++) {
```

```
            print (arr[i][j]);
```

```
            j = j + 1;
```

```
        for (k=1; k<n; k++) {
```

```
            print (arr[i][j]);
```

```
            i = i + 1;
```

```
        for (k=1; k<n; k++) {
```

```
            print (arr[i][j]);
```

```
            j = j - 1;
```

```
        for (k=1; k<n; k++) {
```

```
            print (arr[i][j]);
```

```
            i = i - 1;
```

```
        n = n - 2;
```

```
        i = i + 1, j = j + 1;
```

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

$Tc = O(N^2)$

$Sc = O(1)$

3

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

Constraints

$1 \leq N * M \leq 10^6$

$-10^6 \leq mat[i][j] \leq 10^6$

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

$$TC : O(n \cdot m)$$

$SC: O(1)$

→ return True

|   | 0   | 1  | 2  | 3  | 4  | 5  | $k=10$                                          |
|---|-----|----|----|----|----|----|-------------------------------------------------|
| 0 | -10 | -5 | -2 | 2  | 4  | 7  | $< 10$                                          |
| 1 | -7  | -4 | -1 | 3  | 6  | 9  | $9 < 10$                                        |
| 2 | -2  | 3  | 5  | 7  | 11 | 14 | $14 > 10$                                       |
| 3 | 3   | 6  | 8  | 11 | 14 | 17 | $11 > 10$                                       |
| 4 | 7   | 11 | 12 | 15 | 19 | 20 | $12 > 10$                                       |
| 5 | 10  | 14 | 18 | 20 | 24 | 29 | $10 == 10 \rightarrow \text{Ans} = \text{True}$ |

```
boolean matrixSearch (int arr[][], int n, int m)
```

$i=0$  ,  $j=m-1$  //  $i=n-1$  ,  $j=0$  {ToDo}

while ( $i < n$  &&  $j \geq 0$ )

if ( $\text{arr}[i][j] == k$ )

return `tove`:  $TC: O(n+m)$   
 $SC: O(1)$

else if ( $ar[i][j] > k$ )

$j = j - 1$

else if (a

return false;

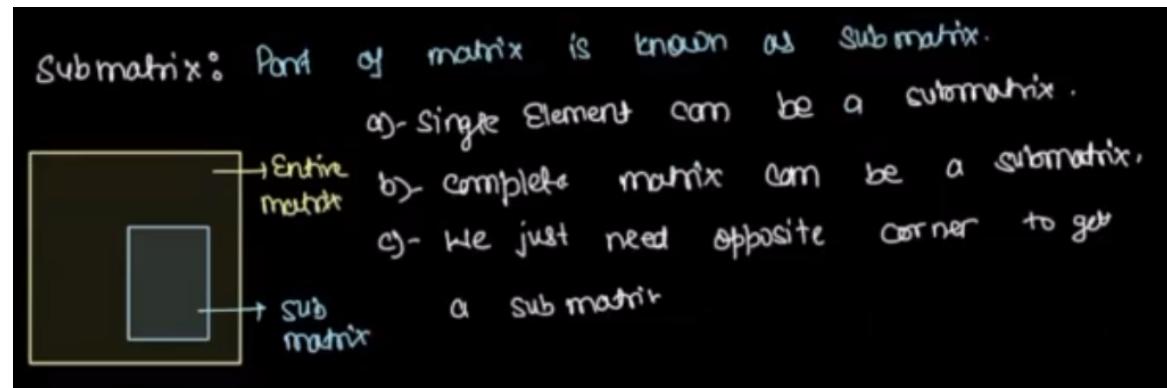
3

# 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



Problem 1: Given a  $n \times m$  matrix and queries, for each query find submatrix sum.

queries

| TL          | BR          | ans |
|-------------|-------------|-----|
| $x_1$ $y_1$ | $x_2$ $y_2$ |     |
| 2 1         | 4 3         |     |
| 3 2         | 5 4         |     |
| 1 2         | 2 3         |     |

|   |    |    |    |    |    |
|---|----|----|----|----|----|
| 0 | 1  | 2  | 3  | 4  |    |
| 0 | 7  | 1  | -6 | 3  | 12 |
| 1 | 10 | 5  | -1 | 0  | 9  |
| 2 | 6  | 4  | -2 | 8  | 11 |
| 3 | 13 | -8 | -5 | 12 | 4  |
| 4 | 3  | 2  | 1  | 9  | 8  |
| 5 | 4  | 3  | -2 | 6  | 2  |

Brute force -> Pseudo code

Pseudo code:

```
for( go to every query) { } q times
     $\underbrace{x_1, y_1}_{TL}, \underbrace{x_2, y_2}_{BR}$ 
    int sum=0;
    for(int i=x1; i<x2; i++) {
        for(int j=y1; j<y2; j++) {
            sum += mat[i][j];
        }
    }
    cout<< sum;
}
```

N → no. of Rows in matrix  
M → no. of columns in matrix  
Q → no. of queries  
T.C.:  $O(q * n * m)$

ii)- Optimised : prefix sum

1D array at  $i^{th}$  index

$psum[i]$   $\Rightarrow$  sum of all elements from 0 to  $i$ -index.

2D array at  $(i,j)$  index

$psum[i][j]$   $\Rightarrow$  sum of all elements in submatrix having  
TL = (0,0) & BR is  $(i,j)$

| A <sub>0</sub> | 0  | 1 | 2 | 3 |
|----------------|----|---|---|---|
| 0              | 3  | 2 | 4 | 1 |
| 1              | -1 | 4 | 3 | 2 |
| 2              | 2  | 7 | 6 | 3 |

| A <sub>0</sub> | 0  | 1  | 2  | 3  |
|----------------|----|----|----|----|
| 0              | 3  | 5  | 9  | 10 |
| 1              | -1 | 8  | 15 | 18 |
| 2              | 2  | 17 | 20 | 26 |

How to generate prefix sum matrix using given matrix:

| A <sub>0</sub> | 0  | 1 | 2 | 3 |
|----------------|----|---|---|---|
| 0              | 3  | 2 | 4 | 1 |
| 1              | -1 | 4 | 3 | 2 |
| 2              | 2  | 7 | 6 | 3 |

1. Row wise prefix sum

| A <sub>0</sub> | 0  | 1 | 2  | 3  |
|----------------|----|---|----|----|
| 0              | 3  | 5 | 9  | 10 |
| 1              | -1 | 2 | 6  | 8  |
| 2              | 2  | 9 | 15 | 18 |

2. Column wise prefix sum

| A <sub>0</sub> | 0 | 1  | 2  | 3  |
|----------------|---|----|----|----|
| 0              | 3 | 5  | 9  | 10 |
| 1              | 2 | 8  | 15 | 18 |
| 2              | 4 | 17 | 20 | 26 |

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

prefix sum matrix:  $\rightarrow$

|   | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 |   |   |   |   |   |
| 1 |   |   |   |   |   |
| 2 |   |   |   |   |   |
| 3 |   |   |   |   |   |
| 4 |   |   |   |   |   |
| 5 |   |   |   |   |   |

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

$$psum[4][3] = P_1 + P_2 + \text{Shade Region}$$

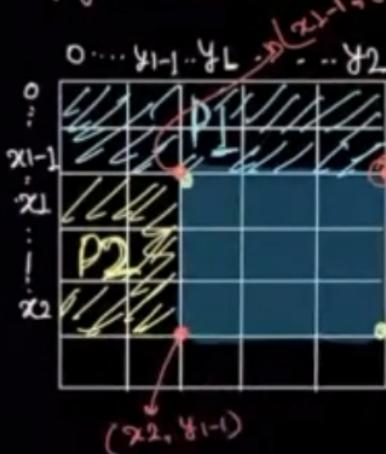
$$\text{Shade Region} = psum[4][3] - \text{Point 1} - \text{Point 2}$$

$$\text{Point 1} = psum[1][3]$$

$$\text{Point 2} = psum[4][1] - psum[1][1]$$

$$\text{Shaded Region} = psum[4][3] - (psum[1][3]) - (psum[4][1] - psum[1][1])$$

prefix sum method:



T.L.

B.Q.

$(x_1, \alpha_1)$

$$(x_2, y_2)$$

$$psum[x_2][y_2] = p1 + p2 + shade$$

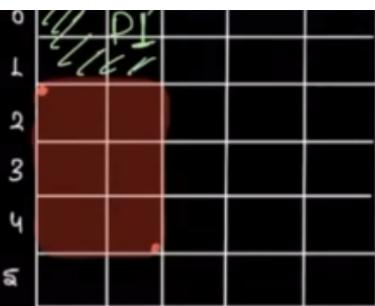
$$\text{Shade} = \text{psum}[(x_2)^T y_2] - p_1 - p_2$$

$$P_1 = \text{psim}[x_1 - \square[y_2]$$

$$P_2 = \text{psum}[x_2][y_{1-1}] - \text{psum}[x_{L-1}][y_{1-1}]$$

$$\text{Sum for 8 submatrix} = p[x_0][y_2] - p[x_1][y_2] - (p[x_2][y_1] - p[x_1][y_1])$$

$(x_1, y_1)$  to  $(x_2, y_2)$



(2,0) SCALER 10

(x<sub>1</sub>, y<sub>1</sub>)      (x<sub>2</sub>, y<sub>2</sub>)

$$\text{Sum} = \underbrace{p[4][1] - p[1][1]}_{\rightarrow \text{ under out!}} + p[4][-1] + p[1][-1]$$

psum[4]fi) = first shade

$$\begin{aligned} \text{shade} &= \text{psum}[4][1] - p_1 \quad \downarrow \\ &= \text{psum}[4][1] - \underline{\text{psum}[1][1]} \quad \nearrow \end{aligned}$$

*observation:*

# of index is out of bound  
don't consider that part in sum.

If index is in minus then we have to apply condition  $x_1 > 0$  and  $y_1 > 0$

|   | 0  | 1 | 2 | 3 | 4 |
|---|----|---|---|---|---|
| 0 | 3  | 2 | 4 | 1 | 6 |
| 1 | -1 | 4 | 3 | 2 | 4 |
| 2 | 2  | 7 | 6 | 3 | 2 |
| 3 | 1  | 2 | 7 | 8 | 1 |

### Prefix Sm

|   | 0 | 1    | 2  | 3  | 4    |
|---|---|------|----|----|------|
| 0 | 2 | (5)  | 9  | 10 | (16) |
| 1 | 2 | 8    | 15 | 18 | 28   |
| 2 | 4 | (17) | 30 | 36 | (48) |
| 3 | 5 | 20   | 40 | 54 | 67   |

$$\text{sum} = p[x_2][y_2] - p[x_{1-1}][y_2] - p[x_2][y_{1-1}] + p[x_{1-1}][y_{1-1}]$$

$$\chi_1 = \perp$$

2

200

112 - 11

$$S_{\text{max}} = \min_{k=1}^K \left( \frac{\partial \mathcal{L}_k}{\partial \theta_k} \right) = \min_{k=1}^K \left[ \frac{\partial \mathcal{L}_k}{\partial \theta_k} \right]_+ + \min_{k=1}^K \left[ \frac{\partial \mathcal{L}_k}{\partial \theta_k} \right]_-$$

$\text{P}_E = \text{P}_{E1} + \text{P}_{E2} + \text{P}_{E3} + \dots$

$$= p(2|4) - p(0|4) - p(2|i) + p(0|i)$$

$$= 48 - 16 - 17 + 8 =$$

```
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 L print sum.
```

```
        int sum = psum [ x2 ] [ y2 ]; q { TL → (0,1) → }
```

```
        { if ( x1 > 0 ) {
```

```
            { | sum - = psum [ x1 - 1 ] [ y2 ] } x
```

```
x1 = 0
```

```
y1 = 1 →
```

```
x2 = 1 }
```

```
y2 = 3 }
```

```
sum = 17 - psum [ 0 ][ 0 ]
```

```
= 17 - 3 = 14
```

```
        { if ( y1 > 0 ) {
```

```
            { | sum - = psum [ x2 ] [ y1 - 1 ] } -
```

```
            { | }
```

```
            { if ( x1 > 0 && y1 > 0 ) { , }
```

```
            { | sum += psum [ x1 - 1 ] [ y1 - 1 ] } ;
```

```
            { | }
```

```
            { | }
```

```
            { | }
```

```
            { | }
```

```
            { | }
```

```
            { | }
```

```
            { | }
```

```
            { | }
```

```
            { | }
```

```
            { | }
```

```
            { | }
```

```
            { | }
```

```
            { | }
```

```
            { | }
```

```
            { | }
```

```
            { | }
```

```
            { | }
```

```
            { | }
```

```
            { | }
```

```
            { | }
```

```
            { | }
```

```
            { | }
```

|    |   |   |   |   |
|----|---|---|---|---|
| A: | 0 | 1 | 2 | 3 |
| 0  | 2 | 9 | 4 | 8 |
| 1  | 1 | 0 | 3 | 2 |

|       |   |   |    |    |
|-------|---|---|----|----|
| psum: | 0 | 1 | 2  | 3  |
| 0     | 2 | 4 | 8  | 11 |
| 1     | 3 | 5 | 12 | 17 |

```
SOPln ( sum ); → 0[φ]: 14 ↵
```

2

3

## Prefix sum subMatrix

How to write code to generate prefix sum in 2D array

- ① Apply row-wise prefix sum on every row.
- ② Apply column-wise prefix sum on every row.

```
Print[] [] prefixSum2D (int[][] A) {
```

```
    ↘ int n = A.length;
```

```
    ↘ int m = A[0].length;
```

```
    ↘ int[][] psum = new int[n][m];
```

```
= { // 1. Apply Row-wise prefix sum.      n=3  
     m=4
```

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

```
        for (int j=0; j<m; j++) {
```

```
            if (j==0) {
```

```
                | psum[i][j] = A[i][j];      i → 2      ↗ j
```

```
            } else {
```

```
                | psum[i][j] = psum[i][j-1] + A[i][j];
```

```
            }
```

```
    }
```

```
= { // 2. Apply column-wise prefix sum.      i → 2      ↗ j
```

```
    for (int j=0; j<m; j++) {
```

```
        for (int i=1; i<n; i++) {
```

```
            | psum[i][j] = psum[i-1][j] + psum[i][j];
```

```
        }
```

```
    }
```

```
    return psum;
```

```
3
```

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

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

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

## Problem -2 - find maximum submatrix sum

Problem 2: Given row-wise and column wise sorted matrix.  
Find maximum submatrix sum.

|   | 0   | 1   | 2  | 3  |
|---|-----|-----|----|----|
| 0 | -20 | -16 | -4 | 8  |
| 1 | -10 | -8  | 2  | 14 |
| 2 | -1  | 6   | 2  | 20 |
| 3 | 5   | 7   | 28 | 42 |

$\Rightarrow \text{TL} \rightarrow (0,0)$   
 $\text{BR} \rightarrow (3,3)$

Max element in entire  
submatrix: last element  
 $= \text{mat}[n-1][m-1]$

NOTE: If we want to  
maximize sum, we  
have to include

last element.

|   | 0   | 1   | 2  | 3  |
|---|-----|-----|----|----|
| 0 | -20 | -16 | -4 | -1 |
| 1 | -10 | -8  | -2 | 5  |
| 2 | -4  | 2   | 4  | 8  |

$\rightarrow \text{TL} = (1,2)$   
 $\text{BR} = (2,3)$

index of max/last

element =  $n-1, m-1$

Bottom Right =  $(n-1, m-1)$  for  
All possible top left

|   | 0   | 1   | 2   |
|---|-----|-----|-----|
| 0 | -50 | -40 | -30 |
| 1 | -25 | -20 | -15 |
| 2 | -10 | -14 | -3  |

$\rightarrow \text{TL} \rightarrow (2,2)$

$\text{BR} = n-1, m-1$

$\text{TL} \rightarrow$  for  $0,0$  to  $n-1, m-1$   
for every possibility.

|   | 0   | 1   | 2  | 3  |
|---|-----|-----|----|----|
| 0 | -20 | -16 | -4 | -1 |
| 1 | -10 | -8  | -2 | 5  |
| 2 | -4  | 2   | 4  | 8  |

$\text{BR} \Rightarrow 2,3$

|                                         |                                         |                                         |
|-----------------------------------------|-----------------------------------------|-----------------------------------------|
| $\text{Top left} \rightarrow$           | $\text{TL} = \underline{\text{BR}}$     | $\text{TL} = \underline{\text{BR}}$     |
| $0,0 \rightarrow 2,3 \rightarrow s_1$   | $1,0 \rightarrow 2,3 \rightarrow$       | $2,0 \rightarrow 2,3 \rightarrow$       |
| $0,1 \rightarrow 2,3 \rightarrow s_2$   | $1,1 \rightarrow 2,3 \rightarrow$       | $2,1 \rightarrow 2,3 \rightarrow$       |
| $0,2 \rightarrow 2,3 \rightarrow s_3$   | $1,2 \rightarrow 2,3 \rightarrow$       | $2,2 \rightarrow 2,3 \rightarrow$       |
| $0,3 \rightarrow 2,3 \rightarrow \dots$ | $1,3 \rightarrow 2,3 \rightarrow \dots$ | $2,3 \rightarrow 2,3 \rightarrow \dots$ |

ans:  $\max(s_1, s_2, s_3, \dots, s_{..})$

If we have prefix sum 2D array,

T.L. index & B.R. index

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

just Apply formula!

$$\text{sum} = p[x_2][y_2] - p[x_{1-1}][y_2] - p[x_2][y_{1-1}] + p[x_{1-1}][y_{1-1}]$$

Steps for solution:

1. Generate prefix sum for 2D array.

2. B.R. is fixed  $\Rightarrow n-1, m-1$

3. Iterate on all possible T.L. index

4. Calculate sum for TL & BR using psum & maxmize sum.

int solve (int[][] A) {

    int[] psun = prefixSum2D(A);      $\Rightarrow O(n*m)$

    int max = Integer.MIN\_VALUE;

    int n = A.length;

    int m = A[0].length;

// BR index is fixed & it is n,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 are TL & x2 & y2 are BR.

            int sum = psun[x2][y2];

            if(x1>0) {

                sum -= psun[x1-1][y2];

            }

            if(y1>0) {

                sum -= psun[x2][y1-1];

            }

            if(x1>0 & y1>0) {

                sum += psun[x1-1][y1-1];

            }

            max = Math.max(max, sum);

    }

    return max;

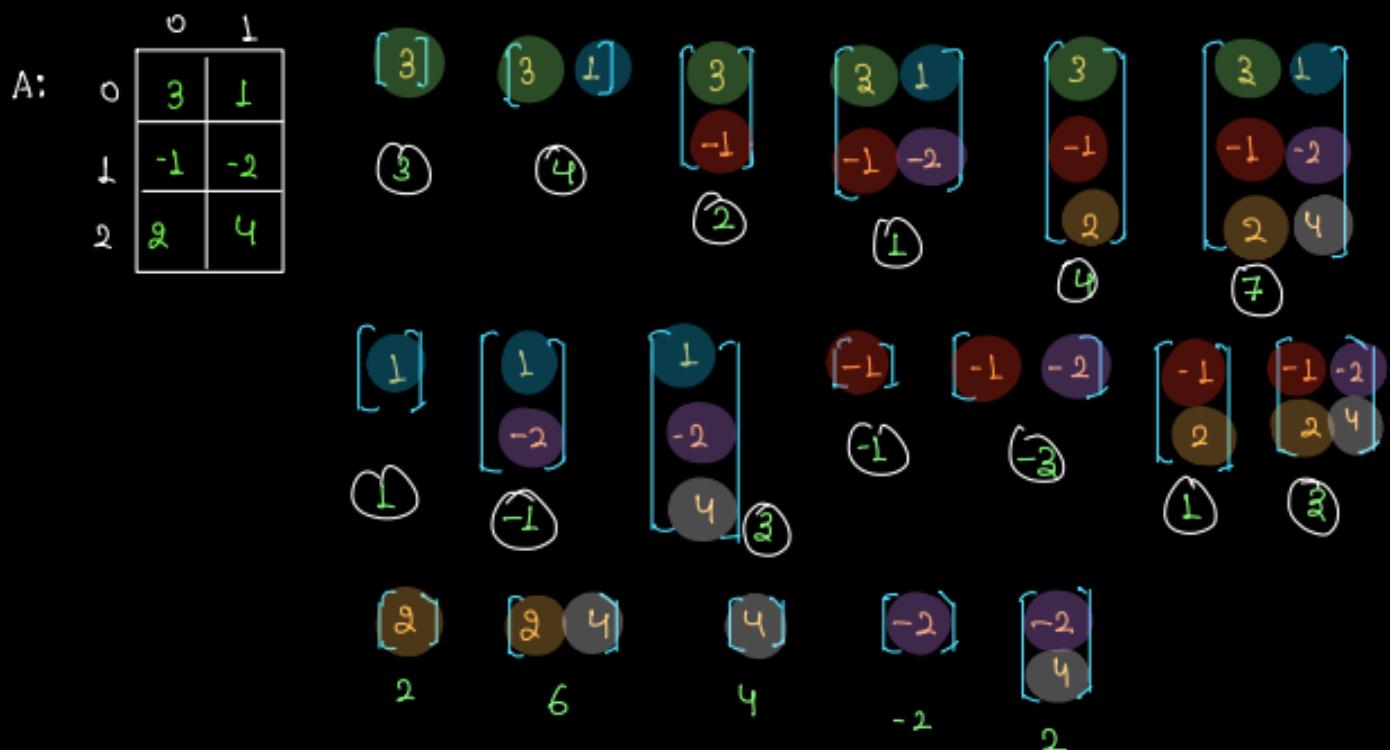
}

$O(n*m)$

T.C.:  $O(n*m)$

S.C.:  $O(n*m)$

Problem 2: Given a  $n \times m$  matrix, find sum of all submatrixes sum.



$$\text{Sum} = 6 * A[0][0] + 6 * A[0][1] + 8 * A[1][0] + 8 * A[1][1] + 6 * A[2][0] + 6 * A[2][1]$$

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

$$= 36$$

$$\text{Conclusion! if } A[i][j] \text{ is coming } n \text{-times in entire sum}$$

$$\text{of all submatrixes, the contribution of } A[i][j] \text{ in}$$

$$\text{sum is } n * A[i][j]$$

|       |       |                     |
|-------|-------|---------------------|
| $m=5$ | $j=2$ | $\frac{m-j}{2} = 3$ |
| 0     | 0     |                     |
| 1     | 1     |                     |
| 2     | 2     |                     |
| 3     | 3     | TL BR BR BR         |
| 4     |       | BR BR BR            |
| 5     |       | BR BR BR            |

$\binom{1,j}{3,2}$

Total possible  
TL index = 12  
total possible  
BR index = 9

Total combination  
by submult =  $12 * 9$   
 $= 108$

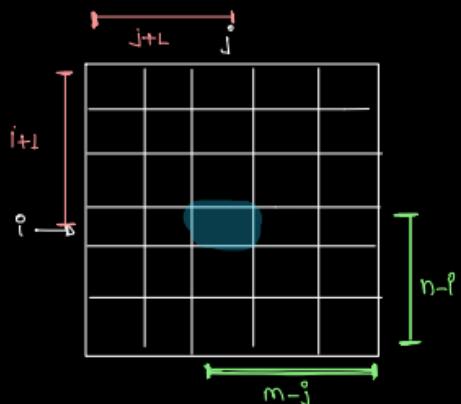
5 Boys      3 girls

|    |    |
|----|----|
| B1 | G1 |
| B2 | G2 |
| B3 | G3 |
| B4 |    |
| B5 |    |

16 possible  
permutation

total Row = n

total col = m



possible top left =  $(i+1) * (j+1)$

possible Bottom Right =  $(n-i) * (m-j)$

contribution 'x' =  $[(i+1)*(j+1)] * [(n-i)*(m-j)]$   
of  $A[i][j]$

```
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];
```

T.C:  $O(n*m)$

S.C =  $O(1)$

```
    }
```

```
}
```

```
return ans;
```

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

1. Decimal Number System:

Base = 10                  digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

$$3849 \Rightarrow 3 * 10^3 + 8 * 10^2 + 4 * 10^1 + 9 * 10^0$$

2. Binary Number System

2. Binary Number system:

Base = 2                  digits: 0, 1

$$\begin{aligned} 1101 &\Rightarrow 1 * 2^3 + 1 * 2^2 + 0 * 2^1 + 1 * 2^0 \\ &\Rightarrow 8 + 4 + 0 + 1 \\ &\Rightarrow 13 \end{aligned}$$

Use index for power

# Binary to Decimal:

①  $(\underline{\underline{1}} \underline{\underline{0}} \underline{\underline{1}} \underline{\underline{1}} \underline{\underline{0}} \underline{\underline{1}})_2 \rightarrow 1 * 2^5 + 0 * 2^4 + 1 * 2^3 + 1 * 2^2 + 0 * 2^1 + 1 * 2^0$

$$\begin{aligned} &= 32 + 0 + 8 + 4 + 0 + 1 \\ &= 45 \end{aligned}$$

## Decimal to Binary

Decimal to Binary:

$$(25)_{10} = (?)_2$$

decimal  
number

|   | 25 | Reminder |        | Result number from Bottom to top |
|---|----|----------|--------|----------------------------------|
| 2 | 12 | → 1      | ↑ top  |                                  |
| 2 | 6  | → 0      |        |                                  |
| 2 | 3  | → 0      |        |                                  |
| 2 | 1  | → 1      |        |                                  |
|   | 0  | → 1      | Bottom |                                  |

## Addition of Binary Numbers

Binary Addition:

|   |   |   |   |           |
|---|---|---|---|-----------|
| 1 | 1 | 1 | 1 | 0 ← carry |
| ↓ | 1 | 0 | 1 | 1         |
|   | 1 | 1 | 1 | 1         |
| — | 1 | 2 | 2 | 3         |
| ↓ | ↓ | ↓ | ↓ | ↓         |
| 1 | 0 | 0 | 1 | 0         |

$$\text{sum} = \text{carry} + d_1 + d_2$$

$$\text{digit} = \text{sum \% base}$$

$$\text{carry} = \text{sum} / \frac{\text{base}}{2}$$

$$\begin{array}{r}
 1 0 1 1 \Rightarrow A=11 \\
 1 1 1 \Rightarrow B=7 \\
 \hline
 1 0 0 1 0
 \end{array}
 \xrightarrow{18} (?)_2$$

|   |   |   |   |   |           |
|---|---|---|---|---|-----------|
| 0 | 1 | 1 | 1 | 1 | 0 ← carry |
| 1 | 0 | 1 | 1 | 0 | 1         |
|   | 1 | 0 | 1 | 1 |           |
| — | 1 | 1 | 2 | 2 | 2         |
| ↓ | ↓ | ↓ | ↓ | ↓ | ↓ ← sum   |
| 1 | 1 | 1 | 0 | 0 | 0         |

$$\text{sum} = \text{carry} + d_1 + d_2$$

$$\text{digit} = \text{sum \% 2}$$

$$\text{carry} = \text{sum} / 2$$

## 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

# Decimal to Binary:

$$(45)_{10} \rightarrow (101101)_2$$

|   |    |     |
|---|----|-----|
| 2 | 45 |     |
| 2 | 22 | → 1 |
| 2 | 11 | → 0 |
| 2 | 5  | → 1 |
| 2 | 2  | → 1 |
| 2 | 1  | → 0 |
| 2 | 0  | → 1 |

top

Bottom

$$(21)_{10} \rightarrow (10101)_2$$

|   |    |     |
|---|----|-----|
| 2 | 21 |     |
| 2 | 10 | → 1 |
| 2 | 5  | → 0 |
| 2 | 2  | → 1 |
| 2 | 1  | → 0 |
| 2 | 0  | → 1 |

# Binary to Decimal:

$$\textcircled{1} \quad (101101)_2 \rightarrow 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 \\ = 32 + 0 + 8 + 4 + 0 + 1 \\ = 45$$

$$\textcircled{2} \quad (101101)_2 = (X)_{10} \quad X = ?$$

$$= 1 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$$

$$= 64 + 0 + \underbrace{16 + 8 + 4}_{+ 0} + 0 + 1$$

$$= 64 + 29$$

$$= 93$$

∴

## # Bitwise operators:

$\&$ ,  $|$ ,  $\wedge$ ,  $\sim$ ,  $<<$ ,  $>>$   
 AND OR XOR NOT left shift right shift

$$\begin{array}{l} \perp | 0 \Rightarrow 1 \\ \perp | 1 \Rightarrow 1 \end{array} \quad \left. \begin{array}{l} \\ \end{array} \right\} \perp \text{ is dominant}$$

$$\left. \begin{array}{l} 0 \& 0 = 0 \\ 1 \& 0 = 0 \\ 1 \& 1 = 1 \end{array} \right\} 0 \text{ is dominant}$$

### AND

Sopln(13  $\&$  10);

$$\perp_0 | p \cdot 8$$

NOTE: 0 is dominant in  
AND

$$\begin{array}{r} 13 \rightarrow [ \perp \perp 0 ] \perp \\ 10 \rightarrow [ \perp 0 ] \perp 0 \\ \hline 1000 = 8 \end{array}$$

OR Sopln(13  $|$  10);

$$\perp_0 | p : 15$$

NOTE:  $\perp$  is dominant  
in OR

$$\begin{array}{r} 13 \rightarrow [ \perp 1 0 ] 1 \\ 10 \rightarrow [ \perp 0 ] \perp 0 \\ \hline 1111 \rightarrow 15 \end{array}$$

XOR Sopln(13  $\wedge$  10);

$$\perp_0 | 7$$

NOTE: Same same puppy  
Sharing

$$\begin{array}{r} 13 \rightarrow [ \perp \perp 0 ] \perp \\ 10 \rightarrow [ \perp 0 ] \perp 0 \\ \hline 0111 \rightarrow 7 \end{array}$$

Same same  $\Rightarrow 0$   
otherwise  $\rightarrow \perp$

left shift:

SOPIn( 15 << 2)

$$a \ll n = a * 2^n$$

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

$$\hookrightarrow (1111)_2$$

Released if  
not

$$\begin{aligned}
 a \ll n &= 15 * 2^2 \\
 15 * 2^2 &= 15 * 4 = 60 \\
 15 &\rightarrow [1 \quad 1 \quad 1 \quad 1]_2 \\
 1+2^5+1+2^4+1+2^3+1+2^2+1+2^1+0+0 &= 60 \\
 32+16+8+4+0 &= 60 \\
 = 40+20+0 &= 60 \\
 = 60 &
 \end{aligned}$$

Bits shifted  
 toward left

9 inserted  
 from  
 right

Right shift:

SOPIn( 200 >> 4);  
↳ op: 12

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

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

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

Right shift by 2

$$\begin{aligned}
 a = 10 &= 8+2 \\
 &= 01010 \\
 1 &\rightarrow 00001
 \end{aligned}$$

$$a \wedge 1 \Rightarrow \underbrace{\begin{array}{cccccc} 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{array}}_{1011}$$

$$\begin{aligned}
 \text{Initially} &\rightarrow [1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0]_2 \\
 &\rightarrow [1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0]_2 \text{ disorder} \\
 \text{After shift} &\rightarrow [1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 1]_2
 \end{aligned}$$

in Right shift

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

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

ON / Set bit → 1  
OFF / unset bit → 0

$n=13, i=2$

$\begin{array}{r} 1 \\ 3 \end{array} \left[ \begin{array}{cccc} 1 & 0 & 1 \\ 2 & 1 & 0 \end{array} \right] \rightarrow \text{true}$

$n=45, i=3$

$\begin{array}{r} 1 \\ 5 \end{array} \left[ \begin{array}{cccc} 1 & 0 & 1 & 1 \\ 3 & 2 & 1 & 0 \end{array} \right] \rightarrow \text{true}$

$n=45, i=4$

$\begin{array}{r} 1 \\ 5 \end{array} \left[ \begin{array}{cccc} 0 & 1 & 1 & 0 & 1 \\ 4 & 3 & 2 & 1 & 0 \end{array} \right] \rightarrow \text{false}$

**Bruteforce -----**

**Idea1: Convert n into binary and check ith bit whether it is on or off**

**Todo**

**Idea2: using AND**

$n = 45$   
 $i = 3$

$45 \rightarrow \begin{array}{r} 1 \\ 5 \end{array} \left[ \begin{array}{ccccc} 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{array} \right]$   
 $1 \ll 3 : \frac{\cancel{0} \quad \cancel{0} \quad \cancel{1} \quad 0 \quad 0}{0 \quad 0 \quad 1 \quad 0 \quad 0} \equiv 1 \ll 3$

$n = 45$   
 $i = 4$

$45 \rightarrow \begin{array}{r} 1 \\ 5 \end{array} \left[ \begin{array}{ccccc} 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \end{array} \right]$   
 $1 \ll 4 \rightarrow \frac{\cancel{0} \quad \cancel{0} \quad 0 \quad 0 \quad 0}{0 \quad 0 \quad 0 \quad 0 \quad 0} \equiv 0$

```
public static boolean checkIthBit(int n, int i) {
    if((n & (1 << i)) != 0) {           45      3
        // 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

|                                                                                                                              |                                                                                                                                                                                                                                                        |                                                                               |
|------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------|
| $45 \rightarrow \text{Binary} \rightarrow$<br>$\begin{array}{r} 45 \\ \times 2 \\ \hline 90 \\ -45 \\ \hline 45 \end{array}$ | $\begin{array}{r} 2   45 & \text{Remainder} \\ \hline 2   22 & \rightarrow 1 \\ \hline 2   11 & \rightarrow 0 \\ \hline 2   5 & \rightarrow 1 \\ \hline 2   2 & \rightarrow 1 \\ \hline 2   1 & \rightarrow 0 \\ \hline 0 & \rightarrow 1 \end{array}$ | 4 times encounter<br>remainder one<br>will be one<br>checking binary of<br>45 |
|------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------|

| <pre>public static int countSetBit1(int n) {     int count = 0;     while(n &gt; 0) {         if(n % 2 == 1) {             count++;         }         n = n / 2;     }     return count; }</pre> <p>time complexity:</p> $n \rightarrow \frac{n}{2} \rightarrow \frac{n}{4} \rightarrow \frac{n}{8} \dots \stackrel{1}{\overbrace{\dots}} \quad O(\log n)$ | $13 \rightarrow \underbrace{8+4+1}_{1 \ 1 \ 0 \ 1}$ | <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>n</th> <th>Remainder</th> <th>Count</th> <th><math>n = n/2</math></th> </tr> </thead> <tbody> <tr> <td>13</td> <td>1</td> <td>1</td> <td>6</td> </tr> <tr> <td>6</td> <td>0</td> <td>1</td> <td>3</td> </tr> <tr> <td>3</td> <td>1</td> <td>2</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>2</td> <td>0</td> </tr> <tr> <td>0</td> <td></td> <td></td> <td></td> </tr> </tbody> </table> <p>loop will stop</p> | n         | Remainder | Count | $n = n/2$ | 13 | 1 | 1 | 6 | 6 | 0 | 1 | 3 | 3 | 1 | 2 | 1 | 1 | 1 | 2 | 0 | 0 |  |  |  |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------|-----------|-------|-----------|----|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|--|--|--|
| n                                                                                                                                                                                                                                                                                                                                                          | Remainder                                           | Count                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | $n = n/2$ |           |       |           |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |  |  |  |
| 13                                                                                                                                                                                                                                                                                                                                                         | 1                                                   | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | 6         |           |       |           |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |  |  |  |
| 6                                                                                                                                                                                                                                                                                                                                                          | 0                                                   | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | 3         |           |       |           |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |  |  |  |
| 3                                                                                                                                                                                                                                                                                                                                                          | 1                                                   | 2                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | 1         |           |       |           |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |  |  |  |
| 1                                                                                                                                                                                                                                                                                                                                                          | 1                                                   | 2                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | 0         |           |       |           |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |  |  |  |
| 0                                                                                                                                                                                                                                                                                                                                                          |                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |           |           |       |           |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |  |  |  |

| <u>Ideas:</u> Number of bits int $\Rightarrow$ 4 byte , 1 byte: 8 bits<br>$\Rightarrow$ 32 bits <p>check every bit if it is on or off</p> <p>from 0 to 31 check every bit in number 'n'</p> <p>if it is ON, increase count</p> <p>otherwise → check next bit.</p> | $n = 45$<br>$\xrightarrow{0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 1 \ 1 \ 0 \ 1 \ 0 \ 1}$<br>$31 \dots 6 \ 5 \ 4 \ 3 \ 2 \ 1 \ 0$ | <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>i</th> <th>True</th> <th>Count</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>False</td> <td>1</td> </tr> <tr> <td>1</td> <td>True</td> <td>1</td> </tr> <tr> <td>2</td> <td>True</td> <td>2</td> </tr> <tr> <td>3</td> <td>True</td> <td>2</td> </tr> <tr> <td>4</td> <td>False</td> <td>2</td> </tr> <tr> <td>5</td> <td>True</td> <td>3</td> </tr> <tr> <td>6</td> <td>False</td> <td>3</td> </tr> <tr> <td>7</td> <td>True</td> <td>4</td> </tr> </tbody> </table> <p>count = 4</p> | i | True | Count | 0 | False | 1 | 1 | True | 1 | 2 | True | 2 | 3 | True | 2 | 4 | False | 2 | 5 | True | 3 | 6 | False | 3 | 7 | True | 4 |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---|------|-------|---|-------|---|---|------|---|---|------|---|---|------|---|---|-------|---|---|------|---|---|-------|---|---|------|---|
| i                                                                                                                                                                                                                                                                 | True                                                                                                                  | Count                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |   |      |       |   |       |   |   |      |   |   |      |   |   |      |   |   |       |   |   |      |   |   |       |   |   |      |   |
| 0                                                                                                                                                                                                                                                                 | False                                                                                                                 | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |   |      |       |   |       |   |   |      |   |   |      |   |   |      |   |   |       |   |   |      |   |   |       |   |   |      |   |
| 1                                                                                                                                                                                                                                                                 | True                                                                                                                  | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |   |      |       |   |       |   |   |      |   |   |      |   |   |      |   |   |       |   |   |      |   |   |       |   |   |      |   |
| 2                                                                                                                                                                                                                                                                 | True                                                                                                                  | 2                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |   |      |       |   |       |   |   |      |   |   |      |   |   |      |   |   |       |   |   |      |   |   |       |   |   |      |   |
| 3                                                                                                                                                                                                                                                                 | True                                                                                                                  | 2                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |   |      |       |   |       |   |   |      |   |   |      |   |   |      |   |   |       |   |   |      |   |   |       |   |   |      |   |
| 4                                                                                                                                                                                                                                                                 | False                                                                                                                 | 2                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |   |      |       |   |       |   |   |      |   |   |      |   |   |      |   |   |       |   |   |      |   |   |       |   |   |      |   |
| 5                                                                                                                                                                                                                                                                 | True                                                                                                                  | 3                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |   |      |       |   |       |   |   |      |   |   |      |   |   |      |   |   |       |   |   |      |   |   |       |   |   |      |   |
| 6                                                                                                                                                                                                                                                                 | False                                                                                                                 | 3                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |   |      |       |   |       |   |   |      |   |   |      |   |   |      |   |   |       |   |   |      |   |   |       |   |   |      |   |
| 7                                                                                                                                                                                                                                                                 | True                                                                                                                  | 4                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |   |      |       |   |       |   |   |      |   |   |      |   |   |      |   |   |       |   |   |      |   |   |       |   |   |      |   |

### Count set bit 1

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

T.C. :  $O(\log n)$

$$n=8 = 2^3$$

$$\text{gtr} = \log_2 2^3 \\ = 3 \text{ times}$$

$$n=16 = 2^4$$

$$\text{gtr}_2 \log_2 16 = \log_2 2^4$$

= 4 times

$$n = 2^{32}$$

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

### Count set bit 2

```
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  
are Iterating

$$n=8 = 2^3$$

$$\hookrightarrow \text{gtr} = 32 \text{ times}$$

$$n=16$$

$$\hookrightarrow \text{gtr} = 32 \text{ times}$$

$$n=2^{32}$$

$$\text{gtr} = 32 \text{ times}$$

## Power of left shift

Power of left shift:

① with AND

→ great help to check if bit is ON or OFF

$$n \& (1 \ll i) != 0 \rightarrow \text{bit is ON}$$

$$n \& (1 \ll i) == 0 \rightarrow \text{bit is OFF}$$

② with OR

$$\begin{array}{r} 45 \longrightarrow \\ 1 \ll 3 \longrightarrow \end{array} \begin{array}{c} \text{OR} \\ \hline \end{array} \begin{array}{c} \boxed{1} \\ \boxed{0} \\ \boxed{1} \\ \boxed{0} \\ \hline \boxed{1} \end{array}$$

$$\begin{array}{r} 45 \longrightarrow \\ 1 \ll 4 \longrightarrow \end{array} \begin{array}{c} \text{OR} \\ \hline \end{array} \begin{array}{c} \boxed{0} \\ \boxed{1} \\ \boxed{0} \\ \boxed{1} \\ \hline \boxed{1} \end{array}$$

Conclusion:  $n | (1 \ll i) \rightarrow$  it will set  $i^{\text{th}}$  bit in number

↳ if Already ON  $\rightarrow$  ON  
 ↳ if OFF  $\rightarrow$  ON

int set  $i^{\text{th}}$  bit (int n, int i){

$$\begin{array}{|l} n = (n | (1 \ll i)); \\ \hline \text{return } n; \end{array}$$

$$\begin{array}{l} n=10 \Rightarrow [ \begin{array}{cccc} 1 & 0 & 1 & 0 \end{array} ] \\ i=2 \quad \underbrace{\ll 2}_{\ll i} \Rightarrow \begin{array}{c} \text{OR} \\ \hline \end{array} \begin{array}{c} \boxed{1} \\ \boxed{0} \\ \boxed{1} \\ \boxed{0} \end{array} \\ \hline \boxed{1} \boxed{1} \boxed{1} \boxed{0} \\ \hline \end{array} \quad \begin{array}{l} 2^{\text{nd}} \text{ bit is} \\ \text{ON now.} \end{array}$$

$$\begin{array}{l} n=10 \quad \not| \\ i=3 \quad \ll 3 \rightarrow \begin{array}{c} \text{OR} \\ \hline \end{array} \begin{array}{c} \boxed{1} \\ \boxed{0} \\ \boxed{1} \\ \boxed{0} \end{array} \\ \hline \boxed{1} \boxed{0} \end{array}$$

unimpaired because  
 $i^{\text{th}}$  bit is  
 already ON

Q. with XOR

$$45 \rightarrow \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ \hline 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

$$1 \ll 3 \rightarrow \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ \hline 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

$$n=45 \rightarrow \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ \hline 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Conclusion: to toggle / flip  $i^{\text{th}}$  bit

$$n = n \wedge (1 \ll i) ; \quad \rightarrow \text{flip } i^{\text{th}} \text{ bit}$$

L if ON  $\rightarrow$  OFF  
L if OFF  $\rightarrow$  ON.

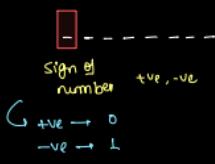
1.  $n \& (1 \ll i)$   $\rightarrow$  check if  $i^{\text{th}}$  bit is ON or OFF
2.  $n | (1 \ll i)$   $\rightarrow$  Set if  $i^{\text{th}}$  bit is ON  $\rightarrow$  ON, OFF  $\rightarrow$  ON.
3.  $n \wedge (1 \ll i)$   $\rightarrow$  Flip  $i^{\text{th}}$  bit, ON  $\rightarrow$  OFF, OFF  $\rightarrow$  ON.

## Negative Number

### Negative Number

8 bits number:

$$\begin{array}{ccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \underline{1} & \underline{1} & \underline{1} & \underline{1} & \underline{1} & \underline{1} & \underline{1} \\ \downarrow & & & & & & \\ +ve, -ve \end{array}$$



0 0 0 0 0 0 0 : +ve 0

~~1 0 0 0 0 0 0~~ : -ve 0

How to store -ve number = 2's complement

2's complement = 1's complement + 1

1's complement = ~ of number.

$$-10 \Rightarrow \underbrace{1's \text{ complement of } 10}_{\text{2's complement of } 10} + 1$$

1's complement of 10  $\Rightarrow$

$$10 \rightarrow \begin{array}{cccccccc} 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ \downarrow & \downarrow \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \end{array}$$

1's complement (010)  $\rightarrow$   $\begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}$  carry from 1  
 $\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$  Binary Addn.

$2's \text{ complement of } 10 = \sim 10 + 1$

$$\begin{array}{r} \sim 10 \\ \hline 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ \downarrow & \downarrow \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \end{array}$$

$$-10 \Rightarrow \begin{array}{|c|} \hline 1 \\ \hline 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ \hline \end{array}$$

Representation

of sign

$$\text{bit. } -1 * 2^7 + 1 * 2^6 + 1 * 2^5 + 1 * 2^4 + 0 * 2^3 + 1 * 2^2 + 1 * 2^1 + 0 * 2^0$$

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

$$= -128 + 118$$

$$= -10 \text{ Ans.}$$

$$-45 = \underbrace{\text{1's complement of }}_{45} + 1$$

$$45 \rightarrow 00101101$$

$$\text{1's complement of } 45 = \sim 45$$

$$\underbrace{\text{2's complement of }}_{\text{Binary.}} 45 = -45 = \sim 45 + 1$$

Binary.

$$45 = 00101101$$

$$\sim 45 = 11010010$$

$$\sim 45 + 1 = \begin{array}{r} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{array}$$

$$\sim 45 + 1 \Rightarrow \begin{array}{r} 2's \\ \text{compl.} \\ 45 \end{array} = -45 = \underline{110100101101}$$

$$\begin{array}{l} \text{Sign} \\ \text{bit} \end{array} \quad \begin{array}{|c|} \hline 1 \\ \hline 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ \hline \end{array}$$

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

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

$$= -128 + 82 = -45 \text{ Ans.}$$

## Range of number

Range of number:  
Sign bit  
8 bit



max : 0 1 1 1 1 1 1 1  
7 6 5 4 3 2 1 0

$$2^0 + 2^1 + 2^2 + 2^3 + 2^4 + 2^5 + 2^6$$

$$\text{sum of GP} = \frac{a[r^t - 1]}{r - 1}$$

$$= \frac{a(r^t - 1)}{r - 1} \quad a=1 \\ r=2 \\ t=7$$

$$= \frac{1[2^7 - 1]}{2 - 1}$$

$$= 2^7 - 1$$

min:



min 1 0 0 0 0 0 0 0  
7 6 5 4 3 2 1 0

$$\text{min} = -2^7$$

Range of number in 8 bits

$$\text{min} = -2^7$$

$$\text{max} = 2^7 - 1$$

$$-2^7 \text{ to } 2^7 - 1$$

32 bits

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

$$\max \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ -\frac{1}{2} & 0 & \frac{1}{2} & \frac{1}{2} \\ -\frac{1}{2} & -\frac{1}{2} & 0 & \frac{1}{2} \\ -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & 0 \end{pmatrix}$$

$$= \frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} + \dots + \frac{1}{2^n}$$

$$= \frac{a(\gamma^t - 1)}{\gamma - 1} \quad \begin{matrix} a = 1 \\ \gamma = 2 \\ t = 3 \end{matrix}$$

$$= \frac{1}{2-1} [2^0 - 1] = 2^0 + 2^1 + 2^2 + \dots + 2^{30}$$

$$= \frac{a + ar + ar^2 + \dots + ar^{t-1}}{r^t - 1} = \frac{a(r^t - 1)}{r^t - 1} = a$$

$$\min : \frac{1}{3} \frac{0}{30} \frac{0}{25} \frac{0}{28} \dots \frac{0}{3} \frac{0}{2} \frac{0}{1} \frac{0}{0}$$

$$m_{in}^o = -\frac{31}{2}$$

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

## Summary of ranges

Summary of Ranges:

If number system have 8 bits:

\* 7<sup>th</sup> bit is there which is sign bit

\* max =  $2^7 - 1$

\* min =  $-2^7$

If number system have 32 bits:

\* 31 bit is there which is sign bit

\* max =  $2^{31} - 1$

\* min =  $-2^{31}$

If number system have 64 bits:

\* 63<sup>th</sup> bit is there which is sign bit

\* max =  $2^{63} - 1$

\* min =  $-2^{63}$

How to calculate bits of -ve number

$-a = 2^s \text{ complement of } a$

= 1's complement of  $a + 1$

1's complement =  $\sim a$

int a = ~a + 1 ]

(-ve)

$-a = \sim a + 1$

or

l'scomp + 1

2's comp

# include <climits>

↓ INT\_MIN  
↓ INT\_MAX

Net sum  
~~Int max + Int min~~

# DSA: Subsets & Subsequences 16 - May 2023

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

## - Modular Arithmetic

Modular Arithmetic:

$$a \% m \Rightarrow \text{Remainder when } a \text{ is divided by } m.$$

for e.g.  $10 \% 4$

The diagram shows the division process:  $\overline{4} \overline{\overline{1} \overline{0}} \overline{\overline{2}}$ . The top part is labeled "dividend", the bottom part "divisor", the middle part "quotient", and the bottom right part "remainder".

$$\text{dividend} = (\text{divisor} * \text{quotient}) + \text{Remainder}$$

$$\text{remainder} = \text{dividend} - \underbrace{(\text{divisor} * \text{quotient})}_{\text{greatest multiple of divisor} \leq \text{dividend}}$$

greatest multiple of divisor  $\leq$  dividend

$$\begin{aligned} \textcircled{1} \quad 47 \% 6, \quad \text{remainder} &= 47 - (\text{greatest multiple of } 6 \leq 47) \\ &= 47 - 42 \\ &= 5 \end{aligned}$$

$$\begin{aligned} \textcircled{2} \quad 38 \% 7, \quad \text{rem} &= 38 - (\text{greatest mult. of } 7 \leq 38) \\ &= 38 - 35 \\ &= 3 \end{aligned}$$

$$\begin{aligned} \textcircled{3} \quad -47 \% 6, \quad \text{rem} &= -47 - (\text{greatest multiple of } 6 \leq -47) \\ &= -47 - (-48) \\ &= -47 + 48 = \textcircled{1} \end{aligned}$$

$$\begin{aligned}
 \textcircled{4} \quad -50 \% 6, \quad \text{rem} &= -50 - (\text{greatest mult. of } 6 \leq -50) \\
 &= -50 - (-54) \\
 &= -50 + 54 = 4 \text{ by}
 \end{aligned}$$

$$\begin{aligned}
 \textcircled{5} \quad -43 \% 7, \quad \text{rem} &= -43 - (\text{greatest mult. of } 7 \leq -43) \\
 &= -43 - (-49) \\
 &= -43 + 49 \\
 &= 6 \text{ by}
 \end{aligned}$$

$-3 < -2 < -1 < 0 < 1 < 2 < 3$

$\begin{array}{c} -7 \\ -14 \\ -21 \\ -28 \\ -35 \\ -42 \\ -49 \\ -56 \\ -63 \\ -70 \\ \vdots \\ \vdots \end{array}$ 
 $\left\{ \begin{array}{l} \text{Multiple of 7} \\ \Rightarrow -49 \text{ greatest number.} \end{array} \right.$

NOTE: Remainder is always +ve.

|            | <u>Google</u>   | <u>Python</u>   | <u>Java</u>                        | $\text{if}(\text{rem} < 0) \{$<br>$\quad \quad \quad \text{rem} += \text{divisor};$<br>$\}$                                                                         |
|------------|-----------------|-----------------|------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| $-47 \% 6$ | $\rightarrow 1$ | $\rightarrow 1$ | $\rightarrow -5 + 6 \Rightarrow 1$ |                                                                                                                                                                     |
| $-50 \% 6$ | $\rightarrow 4$ | $\rightarrow 4$ | $\rightarrow -2 + 6 \Rightarrow 4$ |                                                                                                                                                                     |
| $-43 \% 7$ | $\rightarrow 6$ | $\rightarrow 6$ | $\rightarrow -1 + 7 \Rightarrow 6$ | $\text{if}(\text{rem} < 0) \{$<br>$\quad \quad \quad \text{rem} += \text{divisor};$<br>$\}$<br>$\ominus 47 \% 6$<br>$= -47 \% 6 = -5$<br>$-5 + 6 = \textcircled{1}$ |

|                                     | Rem.                                                                                        |                            |
|-------------------------------------|---------------------------------------------------------------------------------------------|----------------------------|
| $\text{ans \% divisor} \Rightarrow$ | $\text{ans \% divisor} \Rightarrow -2 \equiv 4 \text{ by}$                                  | $-2 + 6 = \textcircled{4}$ |
|                                     | $\text{if}(\text{rem} < 0) \{$<br>$\quad \quad \quad \text{rem} += \text{divisor};$<br>$\}$ |                            |

Return answer with  $\text{ans \% } \underbrace{\frac{9}{10+7}}_x \rightarrow \text{Range of ans} \Rightarrow 0 \text{ to } 21$

$\downarrow$

Why we do this?

$\rightarrow$  just to manage the overflow of answer.

|                                                                                                   |                                                                                         |                                                                                                           |
|---------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------|
| $\left[ \begin{matrix} -\infty \\ \text{number} \\ +\infty \end{matrix} \right] \% 8 \Rightarrow$ | <div style="border: 1px solid black; padding: 5px; display: inline-block;">0 to 7</div> | $47 \% 8 \Rightarrow 7$<br>$40 \% 8 \Rightarrow 0$<br>$42 \% 8 \Rightarrow 2$<br>$-45 \% 8 \Rightarrow 3$ |
|---------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------|

|                                                                                  |                                                                                           |
|----------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------|
| $\left[ \begin{matrix} -\infty \\ +\infty \end{matrix} \right] \% p \Rightarrow$ | <div style="border: 1px solid black; padding: 5px; display: inline-block;">0 to p-1</div> |
|----------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------|

Return any in ~~ans % 10^9 + 7~~

int a = 10<sup>5</sup>;

int b = 10<sup>6</sup>;

int c = a \* b;  $\rightarrow 10^5 * 10^6 = 10^{11}$

$\frac{1}{\text{integer}} \text{ overflow}$

## - Properties of modulo

### Properties of Modulo:

$$\textcircled{1} \quad (a + b) \% m = [a \% m + b \% m] \% m$$

Ex:  $a = 6, b = 15, m = 4$

$$\begin{aligned} \text{L.H.S.} &\rightarrow (6 + 15) \% 4 & \text{R.H.S.} &\Rightarrow [6 \% 4 + 15 \% 4] \% 4 \\ &= 21 \% 4 & &= (2 + 3) \% 4 \\ &= \textcircled{1} & &= 5 \% 4 \\ & & &= 1 \end{aligned}$$

$\underbrace{\text{L.H.S.}}_{=} = \text{R.H.S.}$

$$\textcircled{2} \quad (a * b) \% m = [(a \% m) * (b \% m)] \% m$$

$a = 6$   
 $b = 15$   
 $m = 4$

$$\begin{aligned} \text{In L.H.S.} &\rightarrow & \text{In R.H.S.} & \\ &= (6 * 15) \% 4 & &= [(6 \% 4) * (15 \% 4)] \% 4 \\ &= 90 \% 4 & &= (2 * 3) \% 4 \\ &= 2 & &= 6 \% 4 \\ & & &= 2 \end{aligned}$$

$\text{L.H.S.} = \text{R.H.S.}$

$$\text{③ } (a - b) \% m = (a \% m - b \% m + m) \% m$$

$a=6$   
 $b=5$   
 $m=4$

|                                                                                                 |                                                                                                                                           |                                                                                                                               |
|-------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------|
| L.H.S side<br>$\begin{aligned} & (6 - 5) \% 4 \\ &= -1 \% 4 \\ &= 3 \% 4 \\ &= 3 \end{aligned}$ | in R.H.S $\rightarrow$<br>$\begin{aligned} & = (6 \% 4 - 5 \% 4 + 4) \% 4 \\ &= (2 - 1 \% 4 + 4) \% 4 \\ &= 3 \% 4 \\ &= 3 \end{aligned}$ | $a=15$<br>$b=6$<br>$\begin{aligned} & (15 \% 4 - 6 \% 4 + 4) \% 4 \\ &= (3 - 2 \% 4 + 4) \% 4 \\ &= 5 \% 4 = 1 \end{aligned}$ |
|-------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------|

### - calculate $(a^n) \% p$

Problem 1: Given the value of  $a$ ,  $n$  &  $p$ . calculate  $a^n \% p$ .

constraints:

- a)  $1 \leq a \leq 10^9$
- b)  $1 \leq n \leq 10^{15}$
- c)  $1 \leq p \leq 10^9$

Simplifying  $\left\{ \begin{array}{l} a=3, n=4, p=5 \\ = a^n \% p \\ = 3^4 \% 5 = 81 \% 5 = 1 \end{array} \right.$

$- 10^{18} \text{ to } 10^{18}$

$a^n \% p \Rightarrow (10^9)^3 \% 1000$  Range of  $a^n$   
 $= (10^{27}) \% 1000 \quad \left\{ \begin{array}{l} \text{Range of} \\ \text{Ans is} \\ 0 \text{ to } 1000 \end{array} \right.$   
 $\downarrow \text{int } x$   
 $\log x$

Ex:  $a = 10^9$

$n = 2$

$p = 1000$

Range of answer = 0 to 1000

int solve(int a, int n, int p) {

    ans = 1;

    // calculate  $a^n$  using loop

    for(int i = 1; i <= n; i++) {

        ans = (ans \* a) \% p  $\Leftrightarrow$   
 $(A * B) \% M \Leftrightarrow$

        // returning  $a^n \% p$

        return ans % p;

    }

    return (int) ans;

$a \times a \dots \times a$

$\xrightarrow{n \text{ time}}$

$10^9 \leftrightarrow 10^9$

$0 \text{ to } p-1$

$0 \text{ to } p-1$

$(ans \% p * a \% p) \% p$

$(A \% M * B \% M) \% M$

$10^9 * 10^9$

$10^{18}$

$$(a * b) \% m \Rightarrow (a \% m * b \% m) \% m$$

## - Subsequence

**Subsequences:** By Removing 0 or more Elements from seq.

|                             |                |
|-----------------------------|----------------|
| A: { 3 4 2 -1 9 10 }        | <u>Subseq.</u> |
| x x ✓ x ✓ x → { 2, 9 }      |                |
| ✓ x x ✓ x ✓ → { 3, -1, 10 } |                |
| x x x x x x → { }           |                |
| ✓ x x x x x x → { 3 }       |                |

Some examples of subseq. for given array.

continuous Element from i to j ]  $\Rightarrow$  subarray.

Important point about subseq:

- ① continuous selection of element doesn't matter.
- ② order of indexing will matter here.

array A: { 3 2 4 -1 9 10 }  
index → 0 1 2 3 4 5

{ 3, 2, 4, -1 }  $\xrightarrow{P_1}$  ✓  
 $\xrightarrow{P_2}$  ✓

{ 3, 4, -1, 9 }  $\xrightarrow{P_1}$  ✓  
 $\xrightarrow{P_2}$  ✓

{ }  $\rightarrow$  valid subseq.

{ 3, 9, 4 }  $\rightarrow$  Not valid

Because ordering indexing is not correct.

A  $\rightarrow$  { 3, 1, 2 }

All possible Subsequence

0-len [ { } ]

1-len [ { 3 } ]

[ { 1 } ]

[ { 2 } ]

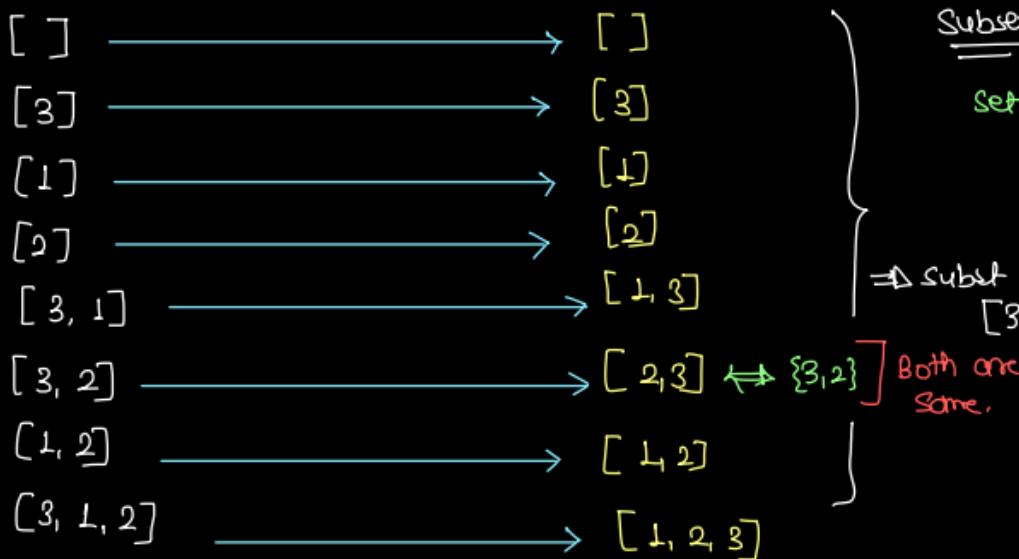
2-len [ { 3, 1 } ]

[ { 3, 2 } ]

[ { 1, 2 } ]

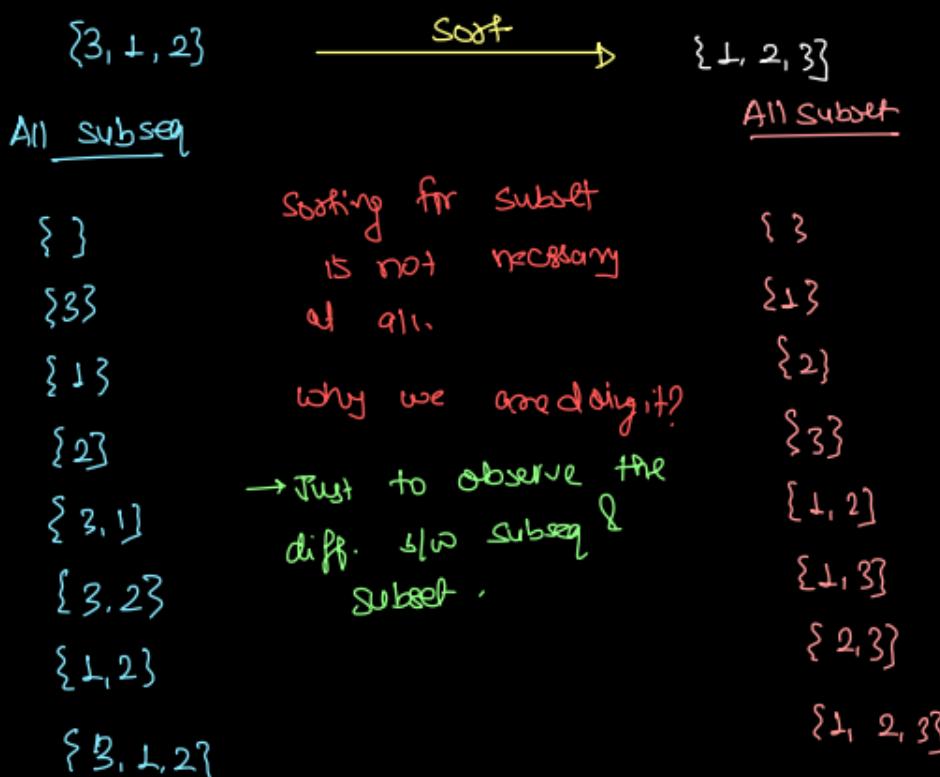
3-len [ { 3, 1, 2 } ]

$A \rightarrow [3, 1, 2] \xrightarrow{\text{Sorting}} [1, 2, 3]$



### - Subset

Subset: same as subseq. but order of indexing doesn't matter here.



#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]  
0 1 2 3  
  
 $\underline{2} \times \underline{2} \times \underline{2} \times \underline{2} = 2^4 = 16$

If length of array is n.

total number of subseq  $\Rightarrow 2^n$

total number of subset  $\Rightarrow 2^n$   
brace under the last two terms

Note:  
 Whenever we deal with subset  
 → Inputs one distinct elemnt.

All the element's one distinct in subset

{L<sub>1</sub>, L<sub>2</sub>}

Subseq. →

All subseq.

{}

{L<sub>1</sub>}

{L<sub>2</sub>}

{L<sub>1</sub>, L<sub>2</sub>}

All subset

{}

{L<sub>1</sub>}

~~{L<sub>1</sub>, L<sub>2</sub>}~~

{L<sub>1</sub>, L<sub>2</sub>}

} 3 subset

Permutation & combination

→ Mathematics

- Sum of Subset is K?

Problem: Given an array [distinct element]. Find if there is any subset with sum = k available or not.

$$A: [2, 7, -1, 5, 6], \quad k=7$$

$$\{7\}, \{2, 5\}, \{-1, 6\} \rightarrow [\underline{\underline{\text{true}}}]$$

A:  $\begin{smallmatrix} 0 & 1 & 2 \\ 3, 2, 4 \end{smallmatrix}$  → How to print all subset.

No of subset  $\Rightarrow 2^3 = 8$

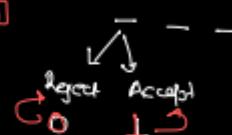
|     | 2 | 1 | 0 |                             |
|-----|---|---|---|-----------------------------|
| 0 → | 0 | 0 | 0 | → $\{\underline{3}\}$       |
| 1 → | 0 | 0 | 1 | → $\{\underline{3}\}$       |
| 2 → | 0 | 1 | 0 | → $\{\underline{2}\}$       |
| 3 → | 0 | 1 | 1 | → $\{\underline{3, 2}\}$    |
| 4 → | 1 | 0 | 0 | → $\{\underline{4}\}$       |
| 5 → | 1 | 0 | 1 | → $\{\underline{3, 4}\}$    |
| 6 → | 1 | 1 | 0 | → $\{\underline{2, 4}\}$    |
| 7 → | 1 | 1 | 1 | → $\{\underline{3, 2, 4}\}$ |



All possible subset

A:  $\begin{smallmatrix} 7 & 4 & 9 \\ 0 & 1 & 2 \end{smallmatrix}$ ,  $n=3$ , no. of subset =  $2^n = 2^3 = 8$

|     | 2 | 1 | 0 | [In bits, Store from Right to Left] |
|-----|---|---|---|-------------------------------------|
| 0 → | 0 | 0 | 0 | → []                                |
| 1 → | 0 | 0 | 1 | → [7]                               |
| 2 → | 0 | 1 | 0 | → [4]                               |
| 3 → | 0 | 1 | 1 | → [7, 4]                            |
| 4 → | 1 | 0 | 0 | → [9]                               |
| 5 → | 1 | 0 | 1 | → [7, 9]                            |
| 6 → | 1 | 1 | 0 | → [4, 9]                            |
| 7 → | 1 | 1 | 1 | → [7, 4, 9]                         |



All subset of array

```

public static boolean checkBit(int n, int i) {
    // use left shift to check if bit is on or off
    if((n & (1 << i)) != 0) {
        return true;
    }
    return false;
}

public static void printSubset(int[] arr) {
    int n = arr.length;
    // calculate  $2^n \Rightarrow 1 \ll n$ 
    int count = 1 << n; //  $2^n$ 
    for(int i = 0; i < count; i++) {
        // for count -> n bits of count
        // bi = bit index
        System.out.print("[");
        for(int bi = 0; bi < n; bi++) {
            // bit is on at bi -> print Otherwise
            if(checkBit(i, bi) == true) {
                System.out.print(arr[bi] + " ");
            } else {
                System.out.print("0");
            }
        }
        System.out.print("]");
        System.out.println();
    }
}

```

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

S.C:  $O(1)$

A:  $[3, 1, 2]$

$n=3$

$$\text{count} = 1 \ll 3 = 2^3 = 8$$

| i       | bit iteration                                            | O/P       |
|---------|----------------------------------------------------------|-----------|
| 0 [000] | 0 1 2<br>0,0 0,1 0,2<br>$\downarrow$<br>number bit index | [ ]       |
| 1 [001] | 0 1 2<br>1,0 1,1 1,2                                     | [ 3 ]     |
| 2 [010] | 0 1 2<br>2,0 2,1 2,2                                     | [ 1 ]     |
| 3 [011] | 0 1 2<br>3,0 3,1 3,2                                     | [ 3 1 ]   |
| 4 [100] | .                                                        | .         |
| 5 [101] | .                                                        | .         |
| 6 [110] | .                                                        | .         |
| 7 [111] | 0 1 2<br>7,0 7,1 7,2                                     | [ 3 1 2 ] |

ToDo task  $\rightarrow$  Dry Run

```

public static boolean checkSumKInSubset(int[] arr, int K) {
    int n = arr.length;
    int count = 1 << n;
    for(int i = 0; i < count; i++) {
        int sum = 0;
        for(int bi = 0; bi < n; bi++) {
            if(checkBit(i, bi) == true) {
                sum += arr[bi];
            }
        }
        if(sum == K) {
            return true;
        }
    }
    return false;
}

```

T.C:  $O(n2^n)$

S.C:  $O(1)$

checkBit

$$n=5 \rightarrow 101101_2$$

$\downarrow \ll 3$

$\frac{101101}{001000}$

$$n \& (1 \ll 3) \neq 0$$

$\rightarrow$  bit is ON

if 0  $\rightarrow$  bit is OFF

## - Sum of Max from every subseq.

Problem 2: Given an array. find sum of max of every subset

A:  $[3, 2, 4]$

$[] \rightarrow x$

$[3] \rightarrow 3$

$[2] \rightarrow 2$

$[4] \rightarrow 4$

$[3, 2] \rightarrow 3$

$[3, 4] \rightarrow 4$

$[2, 4] \rightarrow 4$

$[3, 2, 4] \rightarrow 4$

Ideas 1: using last question Approach

$\rightarrow$  generate all every possible subseq & find max

$\rightarrow$  make eq sum variable and add that max in sum.

T.C:  $O(n2^n)$

S.C:  $O(1)$

Expected T.C =  $O(n \log n)$

Idea 2: using contribution technique

|                                           |                             |                                           |
|-------------------------------------------|-----------------------------|-------------------------------------------|
| $[3, 2, 4]$                               | $\xrightarrow{\text{Sort}}$ | $[2, 3, 4]$                               |
| All possible subseq. max                  |                             | All subset                                |
| $[] \longrightarrow *$                    |                             | $[] \longrightarrow *$                    |
| $[3] \longrightarrow 3$                   |                             | $[2] \longrightarrow 2$                   |
| $[2] \longrightarrow 2$                   |                             | $[3] \longrightarrow 3$                   |
| $[4] \longrightarrow 4$                   |                             | $[4] \longrightarrow 4$                   |
| $[3, 2] \longrightarrow 3$                |                             | $[2, 3] \longrightarrow 3$                |
| $[3, 4] \longrightarrow 4$                |                             | $[2, 4] \longrightarrow 4$                |
| $[2, 4] \longrightarrow 4$                |                             | $[3, 4] \longrightarrow 4$                |
| $[3, 2, 4] \longrightarrow \frac{4}{2^4}$ |                             | $[2, 3, 4] \longrightarrow \frac{4}{2^4}$ |

$$A: [3, 2, 4]$$

Step 1  $\rightarrow$  Sort the array

$$A: [2, 3, 4]$$

2 is max in subset  $\rightarrow 1$  time

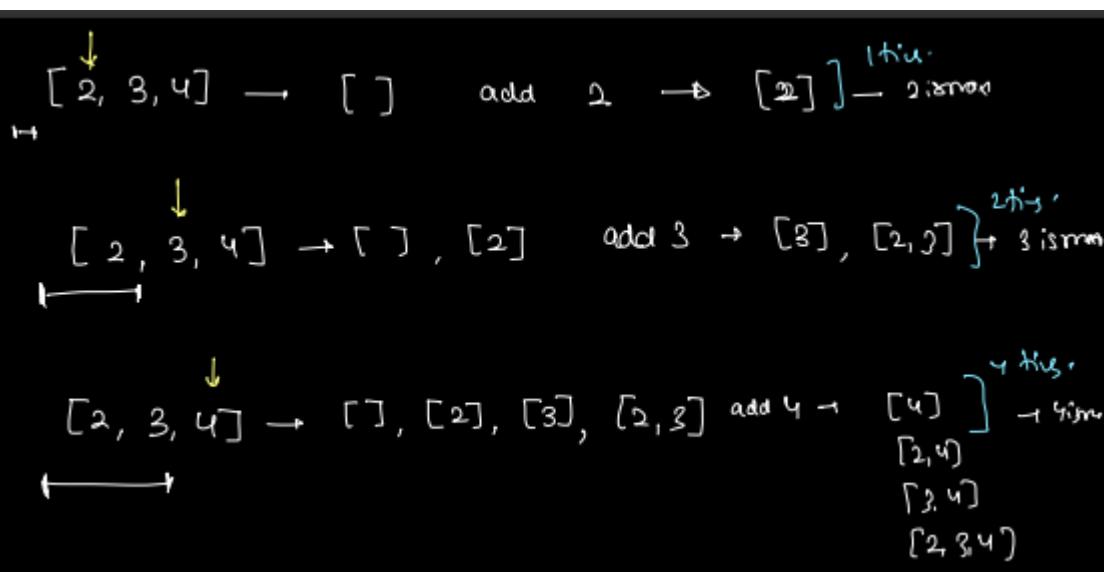
3 is max in subset  $\rightarrow 2$  times

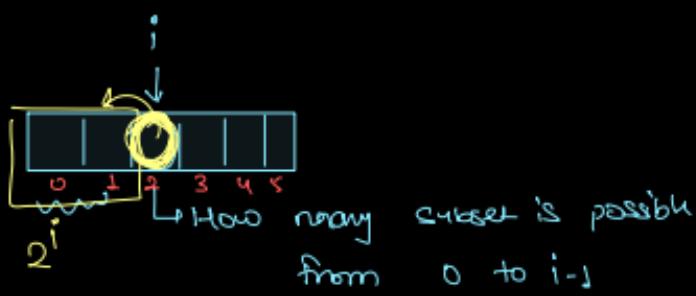
4 is max in subset  $\rightarrow 4$  times

$$\text{Result} = 2 \times 1 + 3 \times 2 + 4 \times 4$$

$$= 2 + 6 + 16$$

$$= \textcircled{24}$$





Element count in left =  $i$

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

[1, 2, 3, 7]

$i=0$   
1 is max

$i=1$   
2 is max

$i=2$   
3 is max

$i=3$   
7 is max

Subset in  
left side  
 $\frac{2^0}{2^1}$   
= 1

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

$\frac{2^2}{2^3}$   
= 4

$\frac{2^3}{2^3}$   
= 8

$$\sum_{i=0}^{n-1} 2^i * \text{count}[i] = 1 + 4 + 12 + 56 = 73$$

int sum(int[] arr) {

ToDo: Manage modulus arithmetic here.

    arraySort(arr);

    —————> n log n

    int sum = 0;

    for (int i=0; i<arr.length; i++) {

Sum += arr[i] \* ( $\frac{\underbrace{1 \dots i}_{2^i}}{2^i}$ )<sup>n</sup>

    return sum;

3

total T.C = n log n + n

Big-O = O(n log n)

S.C = O(1)

# DSA: Contest 2 Discussion - 18 - May 2023

## Flight Bookings

Problem Description :

There are  $B$  flights labeled from 1 to  $B$ .

You are given a 2D array of flight bookings  $A$ , where  $A[i]$  represents a booking for flights numbers from  $A[i][0]$  through  $A[i][1]$  (inclusive) with  $A[i][2]$  seats reserved for each flight in the range.

Return an array of length  $B$ , where each element at index  $i$  is the total number of seats reserved for flight  $i$ .

Note : For example if  $A[i] = \{2, 4, 3\}$  that means in every flight from range 2 to 4, 3 seats are reserved.

Problem Constraints :

$1 \leq |B| \leq 10^5$

$1 \leq |A| \leq 10^5$

$|A[i]| == 3$

$1 \leq A[i][0] \leq A[i][1] \leq n$

$1 \leq A[i][2] \leq 10^4$

Starting flight num.  
Ending flight number  
No. of seats reserved in every flight.

Modeling problem: continuous query sum.

<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.

$B=5$

from third in Seats.

$A :$

|          |          |          |
|----------|----------|----------|
| 1, 2, 10 | 2, 3, 20 | 2, 5, 25 |
|----------|----------|----------|

Flight Number →

| 1  | 2  | 3  | 4  | 5  |
|----|----|----|----|----|
| 10 | 10 | 10 | 10 | 10 |
| 20 | 20 | 20 | 20 | 20 |
| 30 | 30 | 30 | 30 | 30 |
| 40 | 40 | 40 | 40 | 40 |
| 50 | 50 | 50 | 50 | 50 |
| 55 | 55 | 55 | 55 | 55 |

|    |  |    |  |    |  |    |  |    |
|----|--|----|--|----|--|----|--|----|
| 10 |  | 55 |  | 45 |  | 25 |  | 25 |
|----|--|----|--|----|--|----|--|----|

optimised way:

$\text{Index} = \text{Flight No.} - 1$

$B=10$

from  
through  
Seat Resv.  
0 1 2

Flight  
No.

|   | 1  | 2 | 3 | 4   | 5 | 6 | 7   | 8  | 9  | 10 |
|---|----|---|---|-----|---|---|-----|----|----|----|
|   | 0  | 1 | 2 | 3   | 4 | 5 | 6   | 7  | 8  | 9  |
| 0 | 10 | 2 | 2 | -10 | 4 | 5 | -10 | -2 | -7 | 9  |
| 1 | 8  | 0 | 7 | 0   | 7 | 6 | -10 | -2 | -7 | 9  |
| 2 | 0  | 7 | 2 | 0   | 7 | 6 | -10 | -2 | -7 | 9  |
| 3 | 0  | 7 | 2 | 0   | 7 | 6 | -10 | -2 | -7 | 9  |
| 4 | 0  | 7 | 2 | 0   | 7 | 6 | -10 | -2 | -7 | 9  |

A:

|     |   |    |    |
|-----|---|----|----|
| i=0 | 1 | 3  | 10 |
| 1   | 2 | 7  | 3  |
| 2   | 4 | 6  | 10 |
| 3   | 2 | 10 | 5  |
| 4   | 5 | 8  | 7  |

prefix

sum

from arr[s] += val through arr[e+1] -= val Seat Resv.

$\Rightarrow i=0 \quad 1 \quad [ \text{index} = 0 ]$

$3 \quad [ \text{index} = 2 ] \quad 10$

$7 \quad [ \text{index} = 6 ] \quad 2$

$4 \quad [ \text{index} = 3 ] \quad 10$

$6 \quad [ \text{index} = 5 ] \quad 10$

$10 \quad [ \text{index} = 9 ] \quad 2$

$8 \quad [ \text{index} = 7 ] \quad 7$

next  $\rightarrow$  prefix  
sum

$\Rightarrow i=1 \quad 2 \quad [ \text{index} = 1 ]$

$10 \quad [ \text{index} = 9 ] \quad 2$

$\Rightarrow i=4 \quad 5 \quad [ \text{index} = 4 ]$

$8 \quad [ \text{index} = 7 ] \quad 7$

Steps:

① Create arr array of length B having initial elements  
arr[0]: 0.

② Extract query .  
 $s_{\text{start}} = A[i][0]$ ,  $s_{\text{end}} = A[i][1]$ ,  $seats = A[i][2]$

③ For Every query,  
 $ans[s_i] += seats$ ,  
if ( $e_i + 1 < n$ ) {  
 $ans[e_i + 1] -= seats$ .

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

```
public int[] solve(int[][][] A, int B) {
    // 1. Create an array for answer
    int[] ans = new int[B];
    // 2. According to flight booking details,
    // update answer array for every query
    Number of query  $\Rightarrow q$ 
    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
    Number of flight  $\rightarrow n$ 
    for(int i = 1; i < B; i++) {
        ans[i] = ans[i] + ans[i - 1];
    }
    // 4. return answer array
    T.C = O(n)
    Overall T.C
    O(q+n)
    return ans;
}
```

## 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 :

$$A = 3, B = 2$$

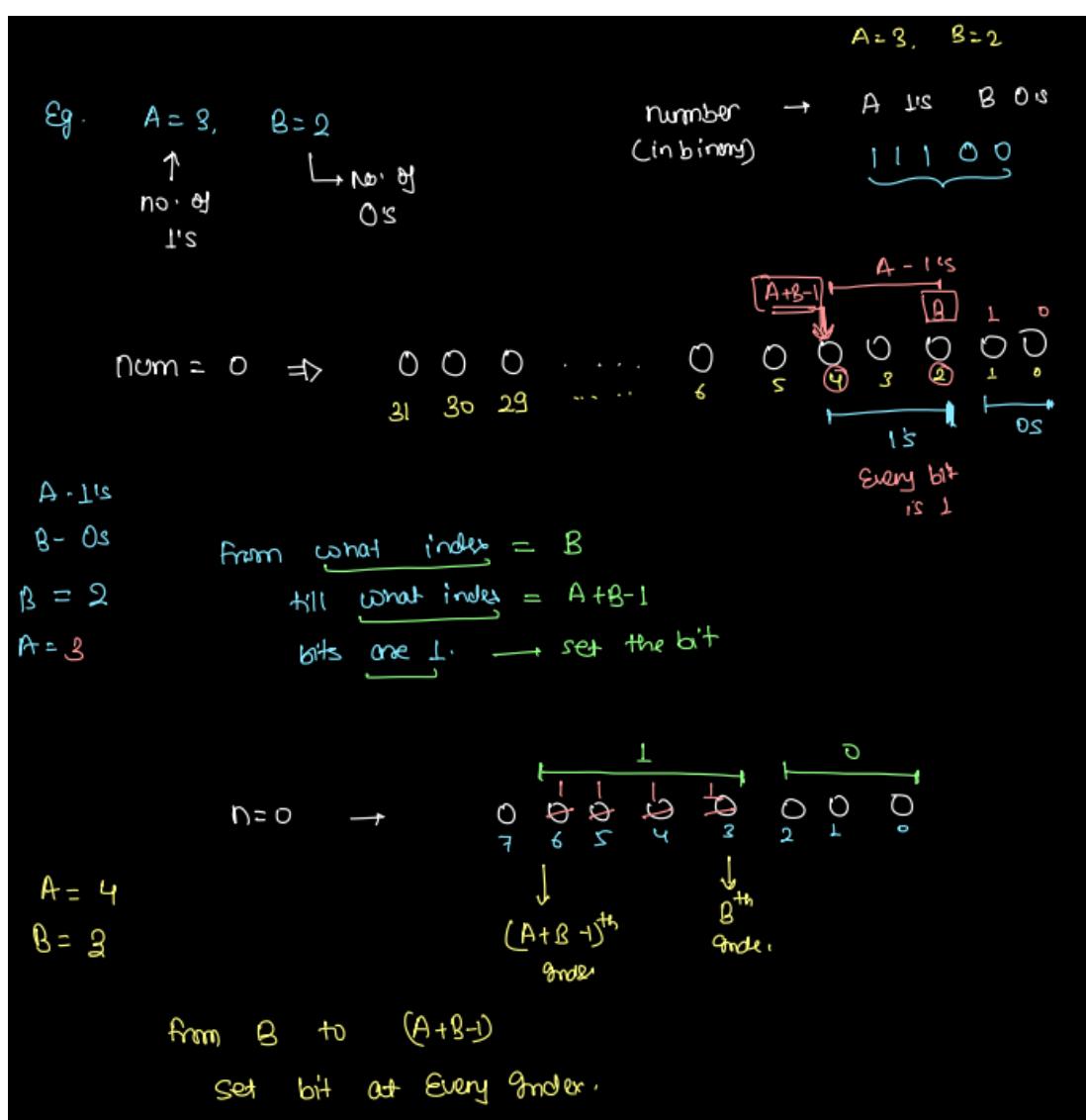
Answer = (11100)2 . Return = 28

#### **Problem Constraints :**

$$1 \leq A + B \leq 30$$

## 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  $i^{\text{th}}$  bit in number

$$n = n \mid (1 \ll 3) =$$

Set 3<sup>rd</sup> index bit

---


$$\begin{array}{r} 1101 \\ 1000 \overline{)11010100} \\ -1000 \end{array}$$

3<sup>rd</sup> bit is 0

How to set  $i^{\text{th}}$  bit in n :

$n = [n \mid (1 \ll i)]$ :

```
int n=0;
```

```
for(Cint i = B; i <= (A+B-1); i++) {
```

```
|     n=(n | (1<<i));  
| }  
| }
```

### maturing

$\begin{matrix} 0 & 0 & 0 & 0 & 0 \\ \textcircled{4} & 3 & \textcircled{2} & 1 & 0 \end{matrix} \leftarrow$   
 A+B-1      B  
 from what index  
 till what index  
 you have to set bit as 1

$$\begin{array}{l} A=4 \\ B=2 \end{array} \quad \text{from } B \underline{\oplus} \quad \begin{array}{l} \\ \text{to } A+B-1 = \underline{(5)} \end{array}$$

```
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; → ansu .  
}
```

D= 0

$$n = \begin{matrix} 5 & 4 & 3 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{matrix} \equiv 0$$

i=2 ( $\perp \Leftarrow \exists$ )

$$n = \begin{array}{ccccccc} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{array} = 4$$

Final

1-3 (1<3)

$$D_1 = \frac{001400}{12} = 12$$

10

$$n = \underline{11110}0$$

j=4 (k<4)

$$n = \begin{pmatrix} 0 & 0 & + & + & - \\ 1 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}$$

originally  $n$  is decimal  
number  $\frac{5}{8}$

15 (1<5)

$$\begin{array}{r} n = \\ 125 \end{array} \begin{array}{r} 0 \ 1 \ 1 \ 1 \ 0 \ 0 \\ \underline{1 \ 0 \ 0 \ 0 \ 0 \ 0} \end{array} = 28$$

## Boundary in Clockwise Direction

### Problem Description

Given a rectangular matrix A of NxM dimension. Return its boundary in clockwise direction.

Problem Constraints :

$1 \leq N, M \leq 10^3$

$1 \leq A[i][j] \leq 10^9$

|   | 0  | 1  | 2  | 3  | 4  | 5  | 6  |
|---|----|----|----|----|----|----|----|
| 0 | 1  | 2  | 3  | 4  | 5  | 6  | 7  |
| 1 | 8  | 9  | 10 | 11 | 12 | 13 | 14 |
| 2 | 15 | 16 | 17 | 18 | 19 | 20 | 21 |
| 3 | 22 | 23 | 24 | 25 | 26 | 27 | 28 |
| 4 | 29 | 30 | 31 | 32 | 33 | 34 | 35 |

ans = 1 2 3 4 5 6 7 14 21 28 35 34 33 32  
31 30 29 22 15 8 .

<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.

Steps : [ N \* M ]

① Iterate on top wall  
Row No = 0 [fixed]  
Col No → 0 to M-1 [iteration]

|   |    |    |    |    |    |    |    |
|---|----|----|----|----|----|----|----|
|   | 0  | 1  | 2  | 3  | 4  | 5  | 6  |
| 0 | 1  | 2  | 3  | 4  | 5  | 6  | 7  |
| 1 | 8  | 9  | 10 | 11 | 12 | 13 | 14 |
| 2 | 15 | 16 | 17 | 18 | 19 | 20 | 21 |
| 3 | 22 | 23 | 24 | 25 | 26 | 27 | 28 |
| 4 | 29 | 30 | 31 | 32 | 33 | 34 | 35 |

② Iterate on right wall  
Col No → M-1 [fixed]  
Row No → 1 to N-1 [iteration]

③ Iterate on bottom wall  
Row No ⇒ N-1 [fixed]  
Col No ⇒ M-2 to 0 [iteration]

④ Iterate on left wall  
Col No ⇒ 0 [fixed]  
Row No ⇒ N-2 to 1 [iteration]

```
public ArrayList<Integer> solve(ArrayList<ArrayList<Integer>> A) {
    int N = A.size();
    int M = A.get(0).size();
    // Dimension N * M
    ArrayList<Integer> ans = new ArrayList<>();
    // 1. Iterate on top wall
    // row -> 0, col -> 0 to M-1
    for(int j = 0; j <= M-1; j++) {
        int val = A.get(0).get(j);
        ans.add(val);
    }
    // 2. Iterate on right wall
    // row -> 1 to N-1, col -> M-1
    for(int i = 1; i <= N-1; i++) {
        int val = A.get(i).get(M-1);
        ans.add(val);
    }
    // 3. Iterate on bottom wall
    // row -> N-1, col -> M-2 to 0
    for(int j = M-2; j >= 0; j--) {
        int val = A.get(N-1).get(j);
        ans.add(val);
    }
    // 4. Iterate on left wall
    // row -> N-2 to 1, col -> 0
    for(int i = N-2; i >= 1; i--) {
        int val = A.get(i).get(0);
        ans.add(val);
    }
    return ans;
}
```

ans → [10, 20, 30, 60, 90, 80, 70, 40]

