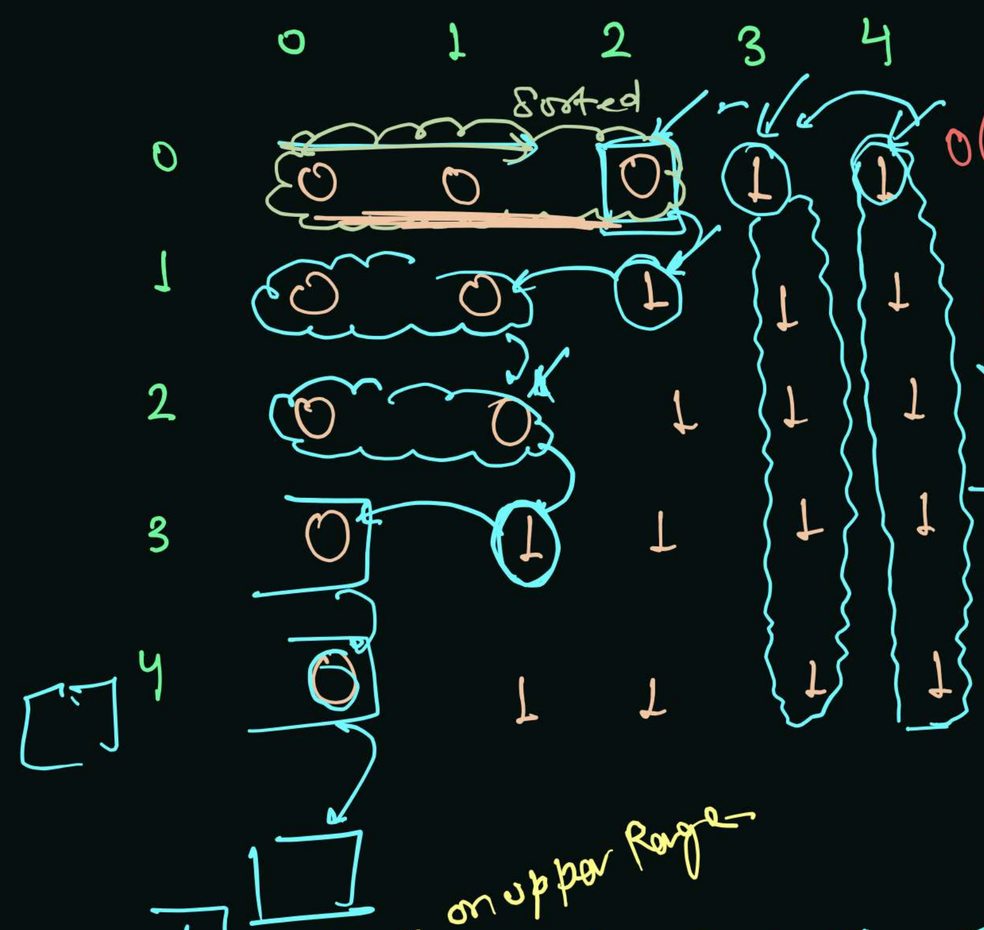
Count Zeros In Sorted Matrix

Thursday, 18 November 2021 8:23 PM

Row-wise, Column wise sorted matrix



matrix[i][j] == 1
j--;

matrix[i][j] == 0
count += (j+1)
i++;

i → Increment → check on upper Range
j → Decrement → check on lower Range

count = 0 1 2 3 4 5

Brute force →

$O(n^2)$ ← 0. Linearly count no. of zeros.

$O(n \log n)$ 1. Binary search on every Row.

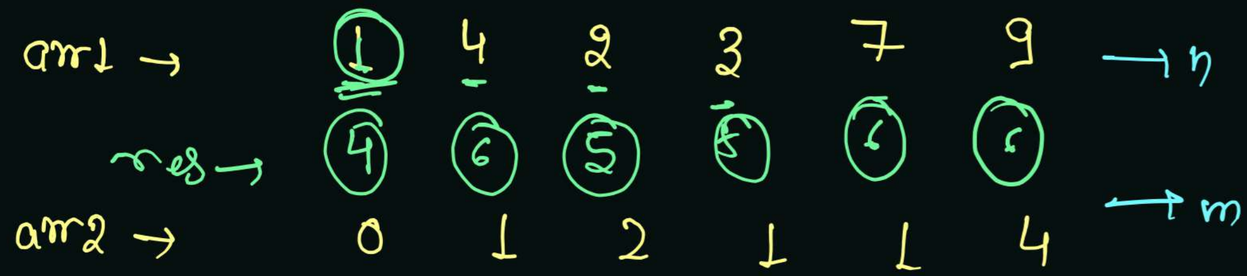
2. Allowed time complexity : $O(n)$

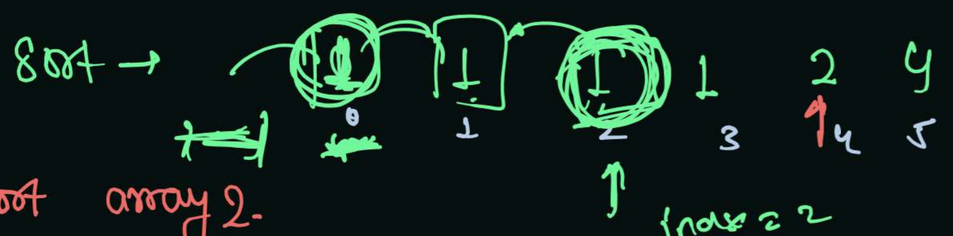
Similar problem:

Hint: Search in a Row wise column wise Sorted matrix.

Counting Elements In Two Arrays

Thursday, 18 November 2021 9:07 PM





Steps → (1) Sort array 2.

(2) travel on array 1 and find ceil index from array 2 where data is equal to arr1[i].

(3) Return res array.

Bruteforce →

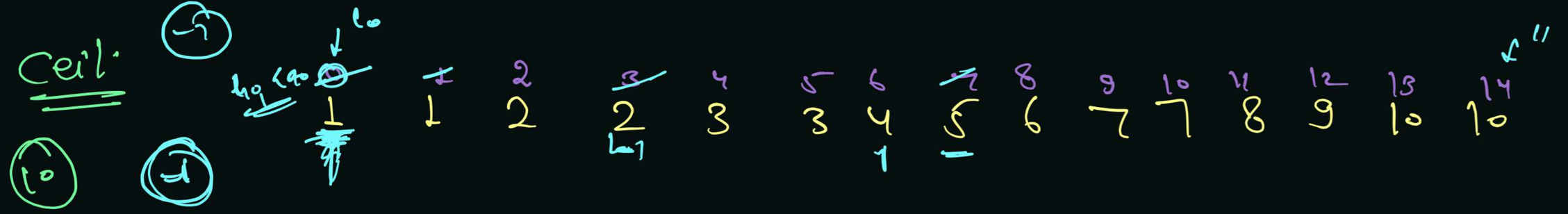
Time → $O(n^2)$

Optimised App. → $O(n \log n)$

Optimised 2 (Space sacrifice) → $O(n)$

time Complex = $O(n \log n)$

Ceil



find ceil Index

find	ceil	Index	floor
1	2	2	1
2	4	5	6
3	6	3	3
7	11	4	5
4	7	10	12
5	8	7	8
9			

data = 1

lo = 0 ~~2~~

hi = ~~14~~ ~~6~~ ~~2~~ 1

mid = $\frac{0+14}{2}$ = ~~7~~ $\frac{0+6}{2}$ = 3 , $\frac{0+2}{2}$ = 1 Idx = ~~7~~ ~~3~~ 2

mid = 2+2 = ②

-1

while (lo <= hi)

```
if( data < arr[mid] ) {
    idx = mid;
    hi = mid - 1;
} else { // data >= arr[mid]
    lo = mid + 1
}
```

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	1	2	2	3	3	4	5	6	7	7	8	9	10	10

lo hi mid hi

floor

data = 10

lo = ~~0~~ ~~8~~ ~~12~~ 13

hi = ~~14~~ 12

mid = $\frac{0+14}{2}$ = (7)

mid = $\frac{8+14}{2}$ = (11)

mid = $\frac{12+14}{2}$ = (13)

mid = $\frac{12+12}{2}$ = (12)

```
while(lo <= hi)
{
    if(arr[mid] < data) {
        indx = mid;
        lo = mid+1;
    } else { // arr[mid] >= data)
        hi = mid-1;
    }
}
```

indx = ~~7~~ ~~11~~ 12

ans

Optimised Approach-

array 1 → 1 2 3 4 7 9

array 2 → 0 1 2 1 1 4

max

max+1

array

Processing-

freq. Array →

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

freq:

freq. less than or equal to (3)

prefix Sum →	1	4	5	5	6	6	6	6	6	7

array:

max

array

freq

prefix Sum

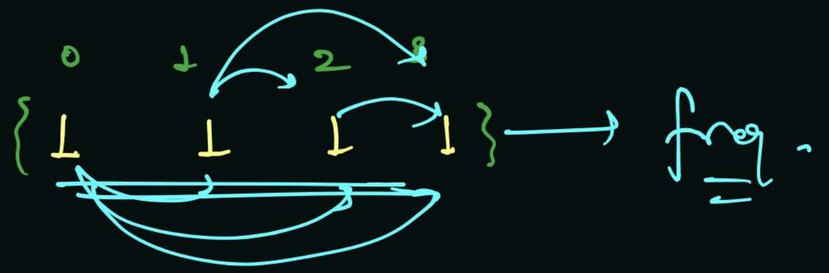
Count Zeros Xor Pairs

Thursday, 18 November 2021 10:37 PM

$arr[i] \wedge arr[j]$

arr[i] XOR arr[j] = 0

$\Rightarrow$ arr[i] = arr[j]



Pairs with index → 0-1 1-2 2-3
0-2 1-3
0-3

4-1$ $\Rightarrow$ No. of pair

= 3 + 2 + 1 = (6)

$^{4}C_{2}$ = (6)

num1 = 10 →

num2 = 10 →

num1 XOR num2 = [0]

5 → 3 times

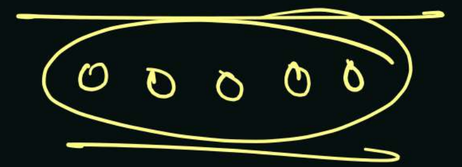
1+2 = (3)

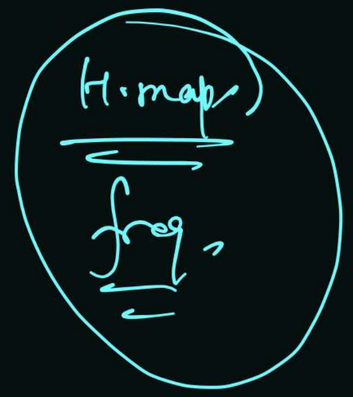
0 → 0 0 0 0 0 (bit)

6 → 0 0 1 1 0

6 → 0 0 1 1 0

6 XOR 6 =





t time freq $\Rightarrow$ pair = $\frac{t(t-1)}{2}$

Array → 1 3 1 3 1 1

HashMap → Element vs freq.

1 → 4

3 → 2

Pair → 1 → (4). $\frac{4 \times 3}{2} = (6)$

3 → (2) $\frac{2 \times 1}{2} = (1)$

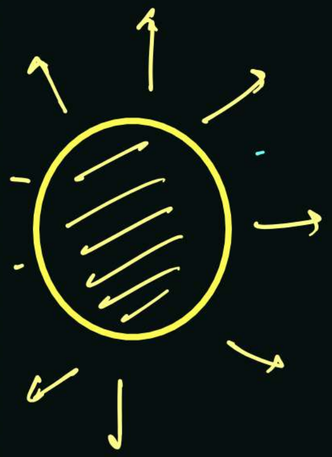
+

[7] Result

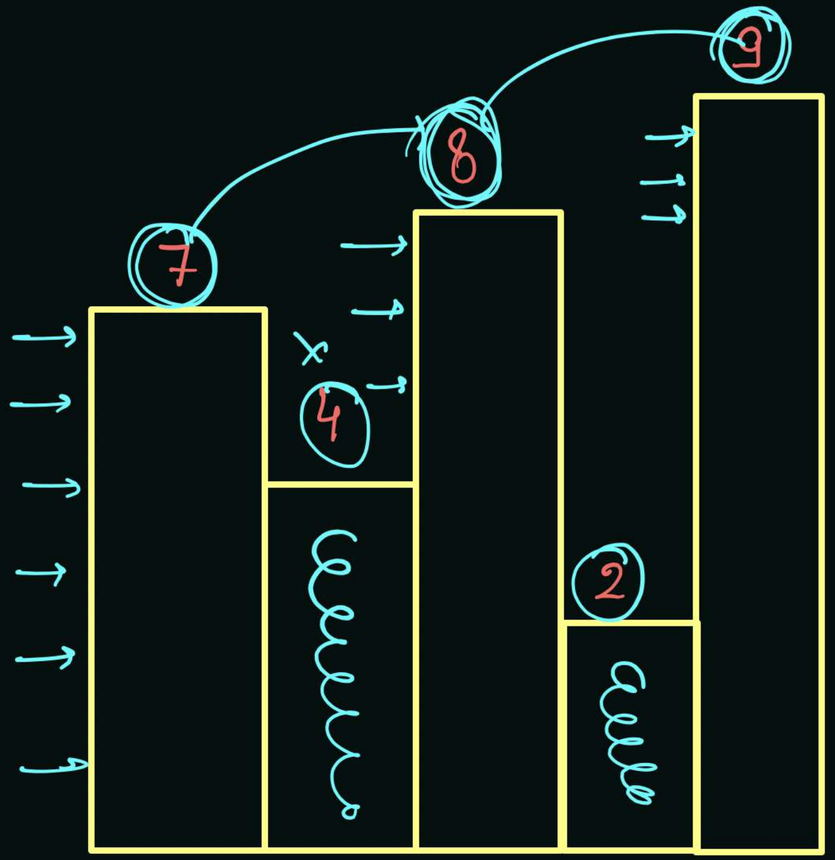
Facing the Sum

Thursday, 18 November 2021 11:18 PM

7 4 8 2 9 = 8



Sun



maintain max

& if max is going to update increment the count.

count = 1 & 3

max = 7 & 9

res

count

Distinct Absolute Array Elements

Thursday, 18 November 2021 11:34 PM

Sorted → Given, to find → distinct absolute Element

array → -3 -3 -2 -2 0 0 1 1 2 2 3 4 4 5 5

Soft

optimised

O(n)

```
while( left <= right)
leftval = abs(arr[left]);
rightval = abs(arr[right]);
if( abs(arr[left]) == abs(arr[right])) {
    if( leftval != prev && rightval != next) count ++;
      prev = leftval;    left ++;
      next = rightval;   right --;
} else if ( abs(arr[left]) > abs(arr[right])) {
    if( leftval != prev) count ++;
    prev = leftval;
    left ++;
} else {
    if( abs(arr[right]) != abs(next)) count ++;
    next = abs(arr[right]);
    right --;
}
```

count = 0 1 2 3 4 5 6

Hashing

Hashing → Hspace O(n)

और - AND

या - OR.