LEETCODE

STRIVER

SHEET

FOLOW SURYAKANT

MORE UPDATE

RAISONI GROUP
a vision beyond

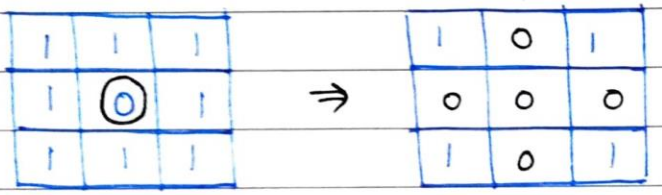
Day - 1

① Set Matrix Zeroes :- [Medium]

Given an 'mxn' integer matrix, if an element is '0' set the entire row and column to ⇒ 0.

You must do it in place :

Input :-



Brute-force approach :-

Whenever we find "0" → then go through its row and make all elements ⇒ -1.
and same, go through column and make all elements ⇒ -1

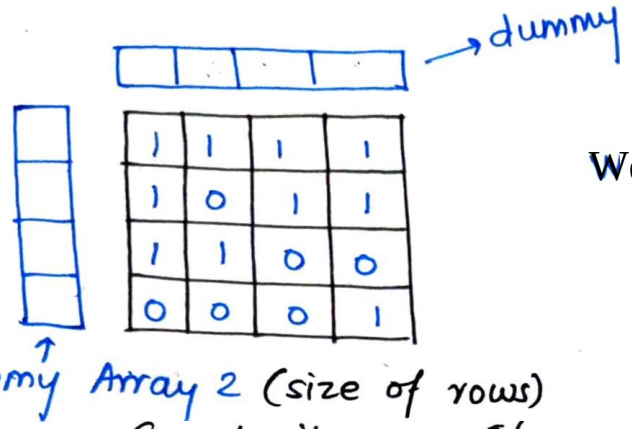
After checking the whole 2D array, then change the -1 element ⇒ to "0"
and here we have got our answer.

Complexity :- (N x m) x (N + M)

Space complexity ⇒ O(1)

(5ptimized approach

(swe O then index



We'lt -fQEe ∩∩ dumm

⇒ -bnearly Complexity ⇒ O(Whene,vem make O ⇒ìn 'the colarnr-)ih

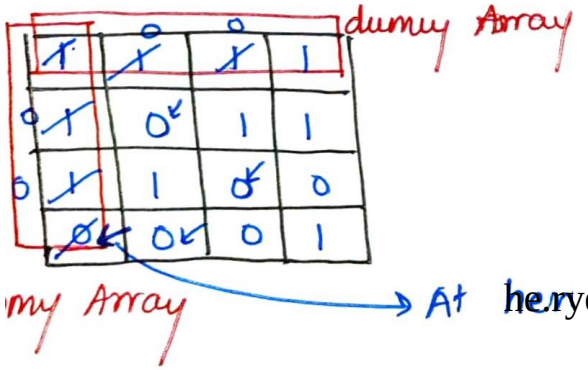
+raveoce Space Complexity ⇒ O in The rr0(.ð-ft' indO< õCNrrn + N km)

OCN) + O (MD

/vfosf O firni ed A roach :-o

—rake neu) vambool

COI Trt.LU



we change ⇒ Co/ =fœue;

check whefhcy for th

Now travese from back

particular element, zero is

Dtmrnr/

dement' zerro is Presen+) n -'he dumm CO/Qm,)

ar•rr-ct dumm DIR) efhen convert ftÌ dement Zero Co).

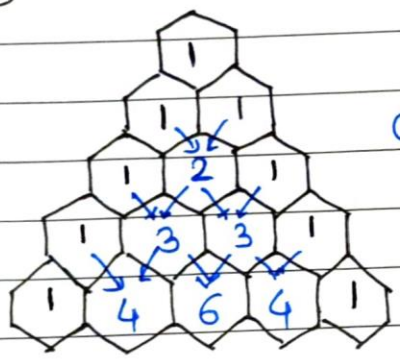
Time complexity = 2*(N x m)

Space Complexity ⇒ O(1)

RAISONI GROUP
— a vision beyond —

② Pascals Triangle :-

Given an integer "numRows", return the first numRows of pascal triangle.



numRows = 5

Output = [[1],
[1,1],
[1,2,1],
[1,3,3,1],
[1,4,6,4,1]]

Approach :-

If in interview, they ask what is the value present at R^{th} row & C^{th} column i.e 5^{th} row & 3^{rd} column ∋ (6)
then formula $(R-1)C_{(C-1)}$

Another Subproblem

print only one of the row from pascals ~~triangle~~
let's suppose 5^{th} row should be printed then

```
for(i=0; i<k ; ++i){
    res *= (n-i);
    res /= (n+i);
}
```

Now for the original problem.
we can resize the vector to (i+1), then for (i=0 ; i<numRows ; ++

```
ans[i][0] = ans[i][i] = 1
```

↓ first column element
↓ last column element

then
for (j=1 ; j<i ; ++j) ⇐

```
ans[i][j] = ans[i-1][j-1] + ans[i-1][j];
```

return ans;

@ Neff Permutafion

ATTthenYs permutations

t 3 t +her) nex+ pemrnaftxtion fir -9

[1,3,2]

- Appomach

Cieneraie All possi ble QBmbos (permutations)

dhen find nums return

ifs next pe7rn

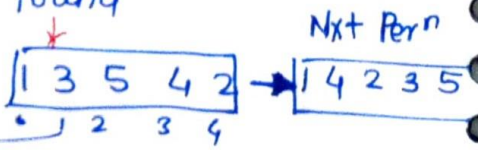
on d

= 2 if la)f nums is Bund -then +he are first permutation
wil he next term

Optimal Approach:-

linearly traverse from backwards
and we have to traverse till we found

arr[i] < arr[i+1]



index 1 = 1

IInd step ~~Again traverse~~

linear traversal from back and find element
which is actually greater than the value at index =1

then we have got 4

index 2 = 3

~~Swap (index1~~

IIIrd step :- Swap(arr[index1], arr[index2])

↳ 1 4 5 3 2

IVth step :-

reverse(index1+1 , last);

Time complexity ⇒ O(n)

21)

(5) Maximum Subarray [Kadane's Algorithm] :-

Given an integer array nums, find the contiguous subarray which has the largest sum. and returns its sum.

Example :-

Input :- nums = [-2, 1, -3, 4, -1, 2, 1, -5, 4]
max. sum = 6

output = 6

Solution :-

```
int mx = INT_MIN ;
```

~~Take one~~ Traverse through array compute sum and take mx = max(sum, mx)
then whenever sum will be < 0
then set sum = 0
return mx.

Time complexity = O(n)

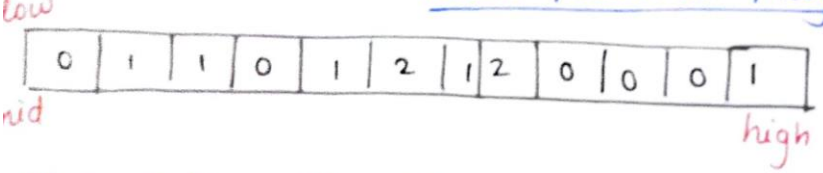
Sort colors [leetcode]

nums' u- array of ')ith n colored obJects as yed, (-ðhzt or blaz soft thern so ±ho-t in-place objeås -fhe sarne enlox arre Qd.jQced uJi+ coJoTs in +he crder Ted, Whife blue

Tnput nums ca to , Q, I[1,0]

oufput – J t 1,2,z)

Approach - This problem can be solved by using Dutch National Flag Algorithm



All the no. from
[0 .. low-1] ⇒ are
[high+1 .. n] ⇒ are

Now shift mid pointer

if mid pointer points ⇒ 0 → then { swap ⇒ (arr[low], arr[mid])
low++;
mid++;

jf mid poin+em
Foi n b

{ swap(arr[mid], arr[high])

Day2

⑦ Rotate Image [Medium]

You are given an ~~2*2~~ nxn 2D matrix representing an Image rotate the image by 90 degree clockwise.

You have to rotate the image in-place, which means you have to modify the input 2D matrix directly.
* Do not allocate another 2D matrix and do the rotation

Example :-

Input ⇒

1	2	3
4	5	6
7	8	9

⇒

7	4	1
8	5	2
9	6	3

Approach :-

Take transpose of the matrix

$A =$

1	2	3
4	5	6
7	8	9

$\Rightarrow A^T =$

1	4	7
2	5	8
3	6	9

then

reverse each row of the transpose matrix

Resultant matrix =

7	4	1
8	5	2
9	6	3

⇐ ANSWER

⑧ Merge Intervals [Me

Given an array of in
merge all overlapping
non-overlapping inte

Otnteïvab end; J

întavcvtð cohere

the

;ntewab and an arr of non. overlÐpìnj
•intervals åhcxt covey the inteyvab (n inrt

Example :-

Intervals : $[[1,3],[2,6],[8,10],[15,18]]$

$[[1,6],[8,10],[15,18]]$

Approach :-

① **Brute-force** sort all the giveo inteyvab.

fien check if the inteyval is mergin yu

T.c o(nJ0 0)

07
nò

⇒ then add to the DS.

A
roach

Q
16
1
13

g
I
H

16117)

1 13

then [1,6] & [8,9] they are not merging ⇒ At this point add [1,6] to our DS.

Now [8,9] ⇒ [8,10]
[8,10]

Again [8,10] ⇒ [8,11]
[9,11]

⇒ [8,11] ✗ [15,18]

Now [15,18] ⇒ [15,17]
[16,17]

Add to OS

sorting

TC = O(NlogN) + O(N) → for traversing

Answer = [1,6], [8,11], [15,17]

-bnea.z¶ ìfera-tL and cheekqœe rney9ìn3 move on next index or not if yes

~~then add~~ in

~~os~~

, Now 1 13

[1,4]

RAISONI GROUP
— a vision beyond —

(9) Merge- Sorted Array in O(1) space :-

You are given two integer arrays 'num1' & 'num2' sorted in non-decreasing order, and two integers m & n. representing the number of elements in num1 & num2 respectively.

Merge num1 & num2 into a single array sorted in non--decreasing order.

Example :-

$a_1[] =$ | 1 | 4 | 7 | 8 | 10 | size = m

$a_2[] =$ | 2 | 3 | 9 | size = n

Approach :-

Take one more array a_3 of size (m+n) and then add a_1 elements & a_2 elements in the a_3 array.

⇓

Sort it

⇓

Take out ~~etc~~ m elements from a_3 put in a_1 then take n elements from a_3 put in a_1

T.C. = O(nlogn) + O(n) + O(n)
≈ O(n)

S.C = O(n)

Approach :- 2

You can't take extra Space

arr1 = 』 7 8

e lin

arr1[i] > arr2[i]

⇓

if yes then

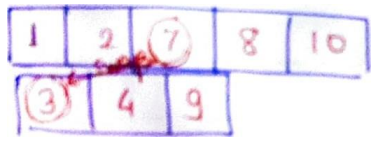
swap(arr2[i], arr2[i])

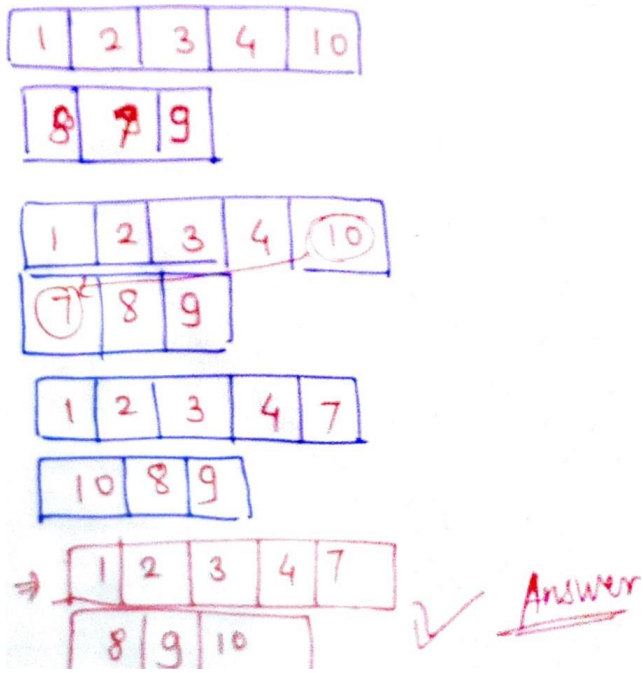
eall in 1 1 0 a 』 if

3

O(n1 x m1)

06)




1 2 3 4 10
8 7 9
1 2 3 4 10
7 8 9
1 2 3 4 7
10 8 9
1 2 3 4 7
8 9 10
Answer

(10) Find the duplicate Number :-

Given an array of integers 'nums' containing 'n+1' integers where each integer is in range [1, n] inclusive.

There is only 'one repeated number' in nums, return its repeated number.

Example :- nums = [1, 3, 4, 2, 2]
output = 2

Approach :-

① sort the nums array,
then we can iterate through nums array

```
for(i=1 to n)
  if(nums[i] == nums[i-1])
      return nums[i];
```

⇒ O(nlogn + n)
space Complexity = O(1)

② Using the frequency array.

Count the frequency of each element in the array & check if the frequency of any element is > 1, return that element

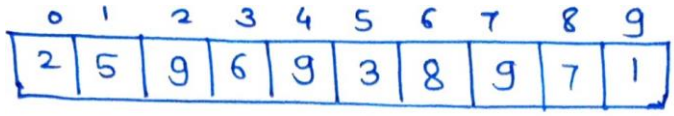
T(n) = O(n)
S(n) = O(n)

```
vector<int> freq(n+1)
for(i=0 to n)
  if(freq[nums[i]] == 0) → freq[nums[i]]++;
  else → return nums[i];
```

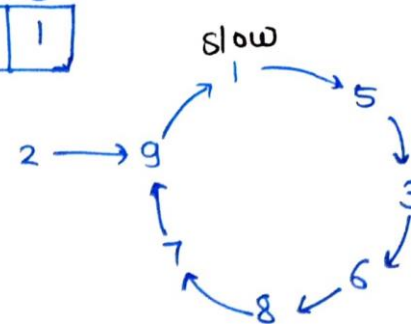
most-

Ôpfirnðe4 method

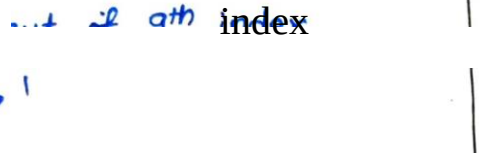
Linked- Ust ûfdL method



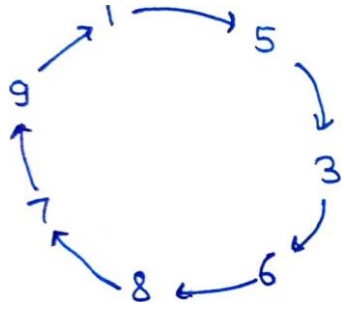
2 +en we
3 [any

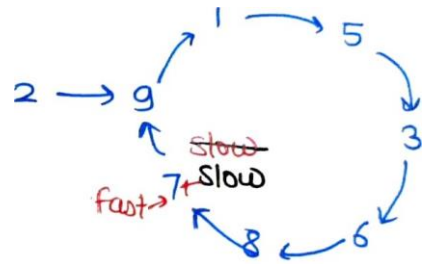


then •take



calce wise





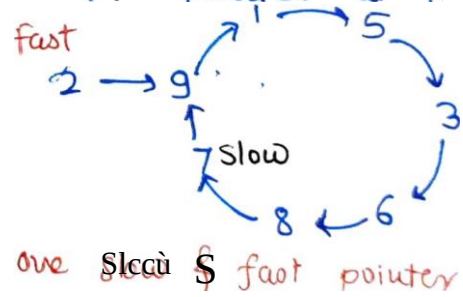
N flow, We will appl ow we will plac -flu

m effi od

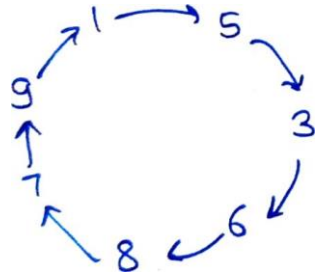
Slow it mova i Step

fQUY

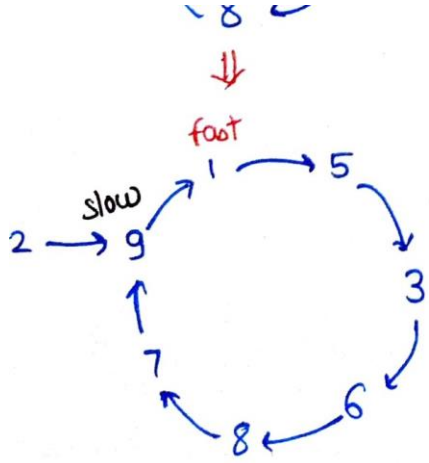
pointerl -3 H mova2 step

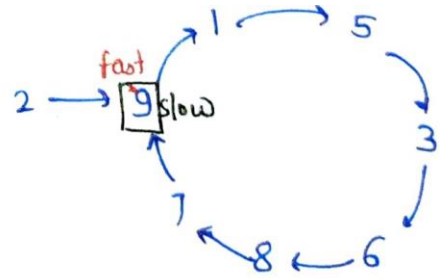


ove Slccù S fast pointer



f0J\





Return 9.

TC = O(N)

SC = O(1)

```
slow = nums[0];
fast = nums[0];

while (slow != fast) {
        slow = nums[slow];
        fast = nums[nums[fast]];
    }
    fast = nums[0];
    while (slow != fast) {
        slow = nums[slow];
        fast = nums[fast];
    }
    return slow;
```

find the repeating (C) and m Pss?n numbers

you aye given an of N integ toìth the valua in -the yan e Cit n) boih incl usive.

Each integerp eQYS exa once except A which appeaTS

B which 61s missinq.

find fie repeating no. missing no .

Emmple tnums = [g, I , 2 t s, 31

ReauLt=

Brute force

Take one substitute Array of size (N+1) and initialize it with 0 then add the values to the substitute array (frequency) and find if the frequency ==0 → missing No.
frequency >1 → Repeating No.

```
for (i=0 to n)
 {  Substitue [Array[i]] ++;
 }
for ( 1 to n)
 {  if (substitute [i] ==0 || Substitute [i] >1)
        {  ans.push_back (i);
        }
   }
```

Time Complexity = O(N) + O(N) ≈ O(N)
Space Complexity = O(N)

RAISONI GROUP
a vision beyond

Approach 2 :-

Using Maths

let the missing no. → x
& the repeating no. → y

S = Sum of Natural no. from 1 to n

$$S = \frac{n(n+1)}{2} \quad \text{——— (I)}$$

P = Sum of Squares of natural no.

$$P = \frac{n(n+1)(2n+1)}{6} \quad \text{——— (II)}$$

Now for Array, also calculate sum → S_1 ——— (III)
& Sum of Squares → P_1 ——— (IV)

then Subtract the

Sum of elements of Array	−	Sum of natural No. from 1 to N

⇓

gives

$$(x-y) = S - S_1 = S'$$

&

$$x^2 - y^2 = P - P_1 = P'$$
$$(x+y)(x-y) = P'$$
$$(x+y) * S' = P'$$
$$x+y = \frac{P'}{S'}$$

Now
$x - y = S'$
$x + y = P'/S'$

$$2x = S'\left(1 + \frac{1}{P'}\right)$$

$$x = \frac{S'}{2}\left(1 + \frac{1}{P'}\right)$$

and find y also

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

(12) Inversion of Array :-

for a given integer array of size 'n' containing all distinct values, find the total no. of inversions that may exist

A pair (arr[i], arr[j]) ⇒ said to be an inversion when

a) arr[i] > arr[j]

and i < j

nums = 3qhexe œre 7

paL.l

(5,1), (5,3), (5,2), (5,4), (3,2), (3,1), (2,1)

4), (3,2), (3,1)

* for the Array which is sorted in decreasing order, the maximum inversions = $\frac{n*(n-1)}{2}$

Approach

Brute force :-

Traverse through array with two loops and check if arr[i] > arr[j] $ $

Co ;

TC = O(n²)

Sc. = O(1)

Day - 03

Search in a Sorted 2D matrix :-

Given a m* p' 2 D matrix iLde exists

(O'Tife a progra.w find the pQ1t•iculQY Tnpu£

:-

matrix = [[1, 3, 5, 7], [10, 11, 16, 20], [23, 30, 34, 60]]

Output =

Appoazh

Brub fbrce:- fra.verse Thro ugh +he 2D matrix if we found the element, rreiuyn Yue.

TC = O(m*n)

sc - oft)

```
for (i=0 ; i < mot.sizeC) 3
        ; j < mat[0].size();
Cjzo    if (mat[i][j] == to
                -return ;
```

F/column =

@ Binary Search t-

3

	0	1	2	3
0	1 (0)	3 (1)	5 (2)	7 (3)
1	10 (4)	11 (5)	16 (6)	20 (7)
2	23 (8)	30 (9)	34 (10)	50 (11)
3	56 (12)	64 (13)	76 (14)	86 (15)

=

go 2

15

mid = 0+19 _ 7 so

So it will present in the right part (20 < target)

low = 8 high = 15

mid = $\frac{8+15}{2}$ = 12 → 56 element

12/4 = 3 → 56 present at (3,0)

12%4 = 0

But our target element present at left part

low = 8 high = 11

mid = $\frac{8+11}{2}$ = $\frac{19}{2}$ = 9 → element = 30

matches with the target element
return true.

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

```
int n = matrix.size();
int m = matrix[0].size();

int low = 0 , high = (n*m) - 1;
while (low <= high) {
    if(matrix[
int mid = (low + (high - low) / 2);
if (matrix[mid/m][mid%m] == target) → return tru
else if (matrix[mid/m][mid%m] < target)
        low = mid + 1;
else → high = mid - 1;

return false;
```

in e

@ Pau) (z, n)

Given a f

ratsed -fo pouJe* • n ba.J;cafff âmp/em POWCztn) ,

.Tnpu£

-X : Q .OOOOO
n - 10

o P 1024.00000

A PPïoocb

U) /3yutL foycz

doa.b/e 1.0

o ; 1 <

ans = ans -z .

me-tuffi ans .

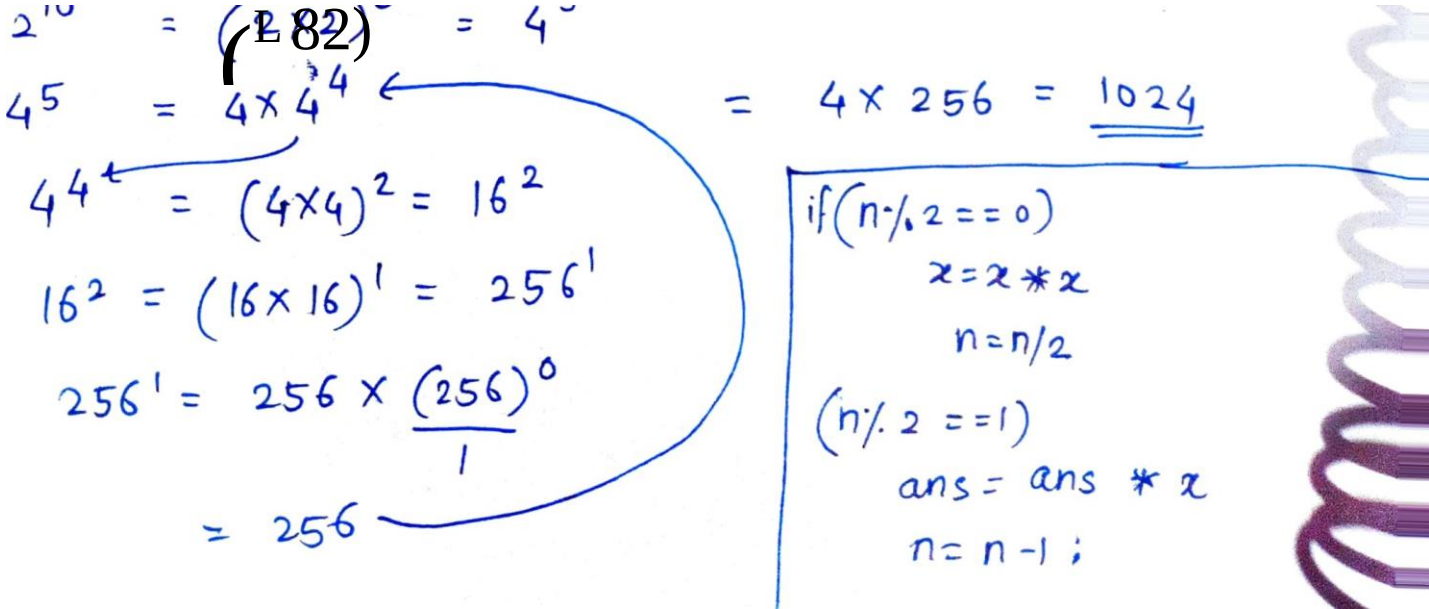
Tme amp lexi= 0

s .c

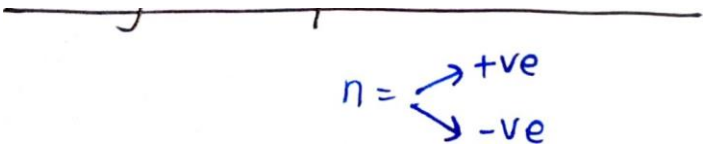
= 0 (

Opfirn"zed A ppTòacb:-

l)sÌn	Bïn	Exponenbiafion



*


η =
+ve
-ve

```
double ans = 1.0
long long nn = n

if (nn < 0)
    nn = -1 * nn;
while (nn > 0)
    if (nn % 2 == 1) {
        ans = ans * x;
        nn = nn - 1;
    }
    else {
        x = x * x;
        nn = nn/2;
    }
}

if (n < 0)
    ans = double(1.0) / (double)(ans);

return ans;
```

T.C = O(logn)

S.C = O(1)

(E) Find the element -that occUTS rnoye -than 112) firne.5

that +han (N (2) N _ 3 nums = c e, 2 (3

errs - I Yim B So

Given o.n of N integeys , a pi an element occuTS mord times

3 O cc no-re fhŒn 3/2 =ãesuL.t = 3

Solution

Q) BY u.ke f0Tce:-

Ch eck the cou-u£ occuyence.8 of a-IL ¿lerne.L.dA

of {he array one from the b one .

S+a-bt dìTSt e(erneni of the anaAd and >)

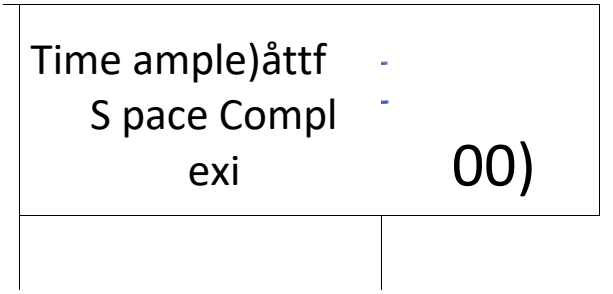
Bun rr•efi-rrn ethQ-Y ebxne" an -f-hL arts-wet-r.

If not ' poceed I-Oi+h -the nex+ elemen-± in+hê amr-a-Lf rep fie Toc-eu .

No. of oCCU-rr€öCL

```
map <int, int> mp
for (int i=0 ; i<arr.size() ; i++)
    mp[arr[i]]++;

for (auto i : mp)
    if( i.second > (n/2))
        return i.first ;
```

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



@ Vsincj Hashrnar:-

Hadhrnaf Ck¶,valuu)
emeut ,

Why this intution worked!

there are 4 partitions

7 7 5 7 5 1 → Majority Ele = minority Element (7 = 3 times, 5 = 2 times, 1 = 1 time; 2+1 = 3)

5 7 → —"—

5 5 7 7 → —"—

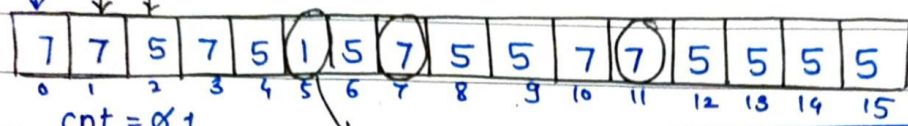
5555 → majority Ele = 5 (4 times appeared)

This is the ans.

③ Most Optimal Solution

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

Moore's Voting Algorithm



n = 16

cnt = ~~0~~ 1

el = ~~0~~ 7

Here count will be zero

```
if (cnt == 0)
    el = a[i];
if (el == a[i])
    cnt++;
else
    cnt--;
```

Now i++; i=1 ⇒ cnt = ~~1~~ 2

a[1] = 7 el = 7

Now i++ ⇒ i=2 ⇒ cnt = ~~2~~ 1

a[2] = 5 el = 7

Now i++; i=3 cnt = ~~1~~ 2

a[3] = 7 el = 7

i++, i=4 cnt = ~~2~~ 1

a[4] = 5 el = 7

i=5, a[5] = 1 cnt = ~~1~~ 0

el = 7

count = 0

i++, i=6 cnt = ~~0~~ 1

a[6] = 5 el = 5

i=7 cnt = ~~1~~ 0

a[7] = 7 el = 5

count = 0

i=8 cnt = ~~0~~ 1

a[8] = 5 el = 5

i=9, a[9] = 5 el = 5

cnt = ~~1~~ 2

i=10, a[10] = 7 cnt = ~~2~~ 1

el = 5

i=11, a[11] = 7 cnt = ~~1~~ 0

el = 5

count = 0

i=12, a[12] = 5 cnt = ~~0~~ 1

el = 5

i=13, a[13] = 5 cnt = ~~1~~ 2

el = 5

i=14, a[14] = 5 cnt = ~~2~~ 3

el = 5

i=15, a[15] = 5 cnt = ~~3~~ 4

el = 5

el = 5 ⇒ This is the answer.

C)

Elements > N [3 times)

Given an N "ntegeys. Pind ihcå a-ppeATs rno-re than (NIB) l±ime.ð -the ATT IF no Such element exists, -return a-n ernph-f vector .ìoac

Q) Brute force :-

Simply COUJI+ The no. of afpeayanc,L each Of eterne,Måa-n usìns nested loops and Whenever you find the COLLLLt- e-temevt greater i-han N/3 +imu elemeÿt will be Yo

```
for Cin} i -o ; i < n i + +) cny= fry C int j = i +1 j<n ; j++)
                cnt+
        ( cnt > CnJ3))
            ans.        -bqck        ;
```

Time complexity = 0 (N 2) SPQC-L complexi 06)

@ Using Hashmap :-

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

unoᵖrderred-rna.p < int, ìnYÞ mp

Qiny ı_o 3 ; i ++)

mp[am[i]]++;

CQL&O :Qmp)

i.second (bt3)) ans.

push_bQck (i.first);

③ Extended Boyer Moore's Voting Algorithm

1	1	1	3	3	2	2	2

num1 = -1
num2 = -1
c1 = 0
c2 = 0

```
vector<int> ans
   count1 = count2 = 0;

if (i=0; i<sz; i++) {
      if (nums[i] == num1)
            count1++;
      else if (nums[i] == num2)
            count2++;
   }
   if (count1 > (sz/3))
      ans.push_back(num1);
   if (count2 > (sz/3))
      ans.push_back(num2);

   return ans;
```

```
for(int el = nums)
{
   if (el == num1) c1++;
  else if (el == num2) c2++;
  else if (c1 == 0)
        num1 = el;
        c1 = 1;
  else if (c2 == 0)
        num2 = el;
        c2 = 1;
  else
        c1--;
        c2--;
```

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

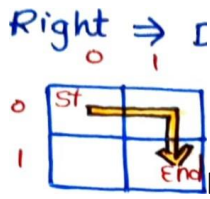
Glivem a matrix , coıxnf pafhs from lQÇt-IDP to -Ihe ai(Jhl böt-tom of Q maiyi x (Diyh -fhe consfrQ7nts -from QQch cem çan eÎlhey önly move İhQ diyecfion ox -to -Yhe aints that nightward AötontDQTd dixec+iöf) .

Ep(arnple Tnput - m: 2 n : 2 ou.tput : 2

GXplonation:-Dot.Dn

⇒m

Tav	



DOU)n

Töto-l @ UJO.YS

.
.-

Hom heye con yo böttom baae Cüne h[+s

OF ENGINQ

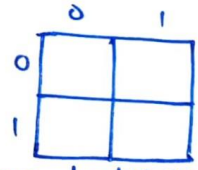
Appyooch

Recursive Approach :-

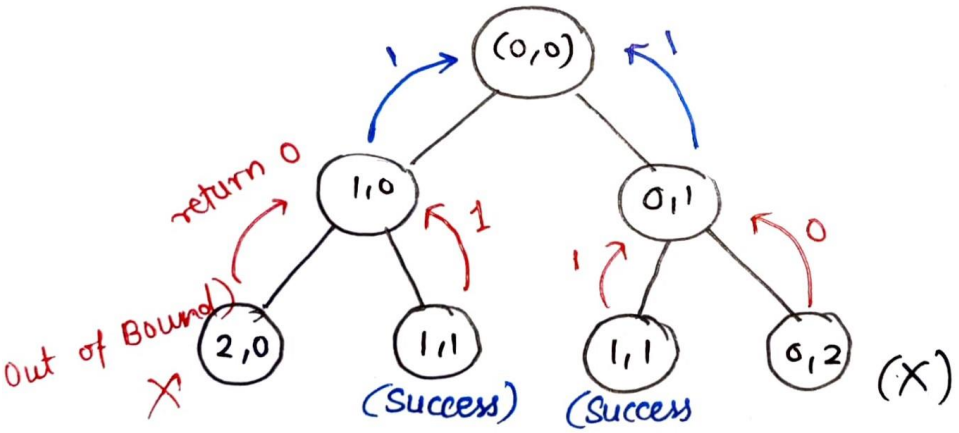
Initially we are at (0,0)

↓

from here we can go to the right as well as bottom



and we will move until the base case hits



Ans = 2

Time & Space Complexity will be exponential

```
countPaths(int i, int j, int n, int m)
{
  if(i==(n-1) && j==(m-1))
      return 1;
  if (i>=n || j>=m) return 0
  else
  return countPaths(i+1, j, n, m)
         +
         countPaths(i, j+1, n, m);
}
```

SO

2

ATS I
Oro

O CN x m)
CC -

0 12

CO co

int

(n -t) -l) -ret-um 1 ;
(i p: n Il • p=m) -re-hufn o ;

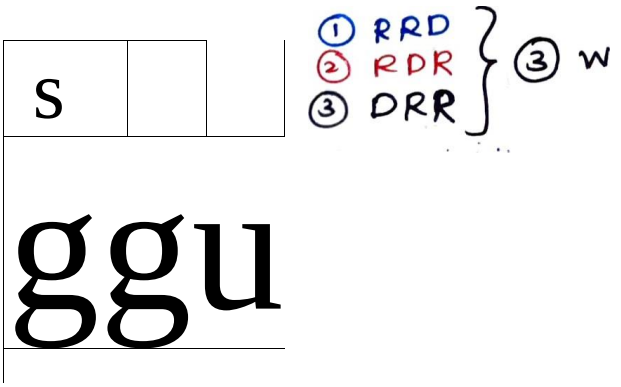
if d yetuyn d
else
-retum d = 1 , j / n / m,

ccutPQ±hs (i j41 m rd p) ;

int (iR.t m t int n)
vec-tDT <vecl-oy > vectoy Cn+1
int num = (Otot p);
Metu n ;

(Ô most Optimal App

Corn binatOMìC8 me±hod



:-

① To -rach -Hte duiina.tian We b hawe -fo EQke nunbepr steps to -the f ceytaîn nuxnberf steps io bottom.

So possible rnova -to +he = m -l possible moveg to the down= n -l

Total ıoves = (m-1 + n-1) rnoVe¿ =

= m+n-2 m

o/e m = 2, n = 3

m+n-2 = 2+3-2 =

m ove fr

¶ìven Exarnp/e

plQces so

if I am able to fill these places $^{m+n-2}C_{m-1}$ (Right path)

or $^{m+n-2}C_{n-1}$ (Bottom paths)

$^{n}C_{r}$

$^{10}C_3 = \frac{8 \times 9 \times 10}{3 \times 2 \times 1}$

T.C = O(m-1) or O(n-1)

S.C = O(1)

2 ;

jvt = double ans

i ,

, i < = risb± ans F

(iota-l

Ciu±) ï)

(18) Count Reverse ~~Paths~~ Pairs :- (HARD)

Given an array of numbers, you need to return the count of reverse pairs.

Reverse pairs :- $arr[i] > 2 * arr[j]$ & $i<j$

Approach :-

① Brute force :-

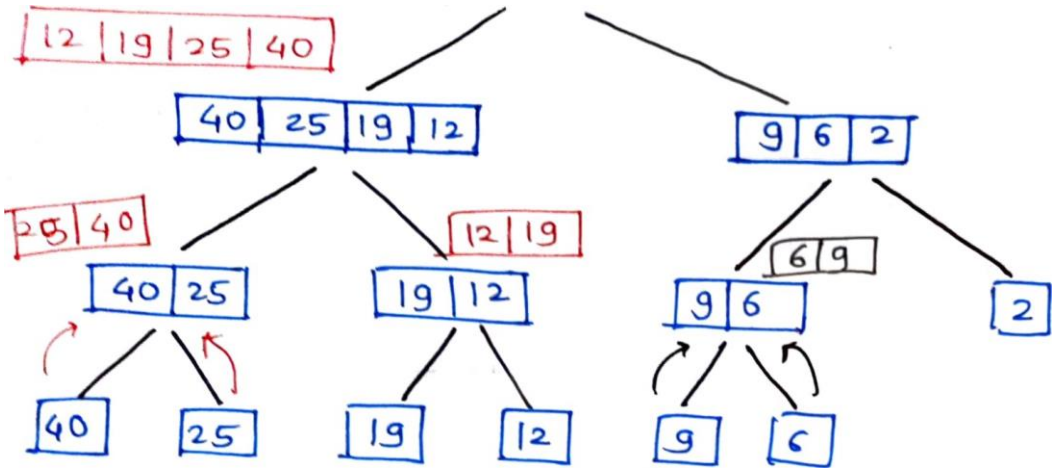
Two nested loops

```
for(i=0 to n)
for(j=i+1 to n)
   if(arr[i] > 2 * arr[j])
        count++;

return count;
```

Time Complexity = $O(N^2)$
Space Complexity = $O(1)$

② Optimal Approach

We will use merge Sort.



Noto we coil/ place ihe 'J ' to such a Index such

-that a ÚJ <=2 * Q Ckeepjn3 consYax.åÒ

over so 2S w nwerf contribuä ouy ans

NOIL) i move fD1vva<d pexfoyrn m

No u)

(D 12

40

(fabe)

25 < = 2*12 move

⇒ This is Quo not contributes to j moVO by one step

so

40 < =) 9

Stop . C right array exhausted)

exhausted

merge operation

ans

⇒ + (2)

arrray

RAISONI GROUP
— a vision beyond —

(i) (j)
9 6 (No contribution)

perform merge step

(i) (j) → j++ (Right array exhausted)
6 9 2

6 <= 2*2 (false) (No contribution) (+1) ans

This is how we have to check.

(i) (j)
6 9 2 (Right array) Exhausted)

and then add total no. of elements on the left so add 1 more to the answer (+1) ans

merge step

(i) (j) (j)
12 19 25 40 2 6 9

12 <= 2*2

move j

12 <= 2*6 (✓) (add no. of elements on left of j) ⇒ (+1)

move i ⇒ to 19

19 <= 2*6 (X)

move j

19 <= 2*9 (X) move j ⇒ Right array exhausted

Add # elements on left of j pointer for (19) ⇒ (+3) ans

move i to 25

25 <= 2*2 (X)
25 <= 2*6 (X)
25 <= 2*9 (X)

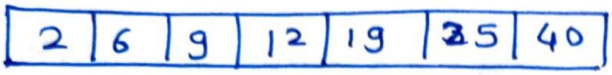
} Right array exhausted Add (+3) ans

so ans = 1+2+1+1+1+3+3+3

ans = 15 ⇒ This is no. of p

poz.¿J

NORJ merge Site



Time Complexity = O(nlogn) + O(n) + O(n)
Merge Sort — merge operation — Count operation

Space Complexity = O(n)
Temp. array in merge sort

int reversePairs (vector <int> & nums)

-yeti-nn mergeS0Nt Cnurns , O, nums .sizeC) - l) ;int merrqeS0Nt C vedÙ1 <iut> &nuns, ìRt Iou), ivå hì9h) , high) return O;

mid -C(ou.) t 'igh)/ 2; int = mergeSortCnurns, JO(.D , mid);inv + = m aseS01t (num.-s t mid + t, hi9h) ; // Right recurs inv -+ = me IF (rums, loco, mid, hi±) →(myetuvn inv .

```
int merge(vector<int>& nums, int low, int mid, int high){
    int cnt=0;  // store the total no. of pairs
    int j=mid+1; // put the j index to the first index of
                                        right half
    for(int i=low; i<=mid; i++){
        while(j<=high && nums[i] > 2LL * nums[j]){
            j++;
        }
        cnt += (j-(mid+1)); // count the # elements placed
                                    to the left side of 'j'
    }
// merge function
    vector<int> temp;
    int left=low, right=mid+1;
    while(left<=mid && right<=high){
        if(nums[left] <= nums[right]){
            temp.push_back(nums[left++]);
        }
        else{
            temp.push_back(nums[right++]);
        }
    }

// if any array exhausted, then:
    while(left<=mid){                          } If left array
        temp.push_back(nums[left++]);          } is left out.
    }

    while(right<=high){
        temp.push_back(nums[right++]);
    }
```

/// Copy back -Hiv '-o nums Qma-I

forCint Iou) high ; i++) { numsÛJ= tempt

/ou.)J•,

3

•eturr) cnt ;

(19) **Two - Sum - Problem**

Given an array of intergers "nums" and an integer "target"
return indices of the two numbers such that they add upto "target".

Example :- Input : nums = [2,7,11,15] , target = 9
output : [0,1]

Solution :-

① Brute force :-

Simply traverse through the nums array

```
for (int i=0 ; i<n ; i++)
  for (int j=i+1 ; j<n ; j++)
      if (num[i] + nums[j] == target) {
          ans.push_back(i);
          ans.push_back(j);
          break;
      }
if (ans.size() == 2) break;
return ans.
```

Time complexity = $O(n^2)$
Space Complexity = $O(1)$

Sori the array.

e wo- oint'ey nppoach

two vcrhable.s eacb Starł from one a[n] d tycweyse in both direch'on

fo quired

find łhe re l elemr each , coe -tY do find dhe Second

'targer - i'

Vecioy ans, temp, temp nams , stoy?

nums array

Soi t (temp) .id' JeĆłzo

n -1;

int x

CLeff Cn3ht) if) {

C tełnpŰ,.lef4J + temp[right]

temp C-IełhJ•, temp break ,

3 else ił (temp[left] + temp

4eft + + ;

x indices of nums array.

/// Pun m one 60p find flu -z

Vr izo ; icnams.sjzeCD ; i + kD i

if (nums[i] == x)

ans.puł) back Ci); i f Cnuzns }

ans. ms[i] == y)

push_back(i);

ans

T.	C —
	OCn log n) ocn)

We'll we The haJhm do See if 'here's _cr vaÜL(tagež dhal can be added it) dhL cunfent axrau| VaĐL CiDHb gcUdhL úml cqu.aa forget

-f the map R)

•ndez index s ocah

uno[r]dered -my ,

foffCi=o. i < n 'H k)

-n m send C)

ayu

n

4 - Sum problem

Given an array of nums.
return an array of unique quadruplets, such that

nums[a] + nums[b] + nums[c] + nums[d] == target

20

Example nums 1, 0, -1, 0, -2, 2] +Qã8Q.t- = O ou-žpu.£
:- C [- -J, 1, 2],
c- 2 t o t o t 2)

- Solutions ⇒

① Using 3 pointers & Binary Search :-

Sort the array Target = 9

Ex :- Given array :- | 4 | 3 | 3 | 4 | 4 | 2 | 1 | 2 | 1 | 1 |

sort it ⇒ | 1 | 1 | 1 | 2 | 2 | 3 | 3 | 4 | 4 | 4 |

i j k

starts
i=0
j=i+1
k=j+1

so nums[i] + nums[j] + nums[k] = 1+1+1 = 3

so we have to find = 9 - 3 = (6) in the right half (using binary search)

'6' is not there in the array. So move k

again element (i+j+k) = 1+1+2 = 4 ⇒ 9-4 = (5) find X

T.C = O(N³log n) + O(n log n)
S.C = O(1)

Now when

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

i j k Do right-half (Binary Searc

nums[i] + nums[j] + nums[k] = 1+1+3 = 5

we have to find = 9 - 5 = (4)

so we have got | 1 | 1 | 3 | 4 | as one quad

push

| 1 | 1 | 3 | 4 |

[-1, 0, 0, 1]

he -right

X

Cßincuy

```
set <        St ;

   FOI ( k cJ 41 to n) {  x : -tQï9Qt -
                (nunsCíJ+ numsCJ+ numstk))
                                be    + k+J nums

                Vectoy < > uad .
                ush- back Gnuns  ;
                uad. ush -back CnurnsCjJ
                uQd. usb- back (nums uad.
                ush back cx) >
                   Sol£C QQd.be in() en
               st. inselt
```

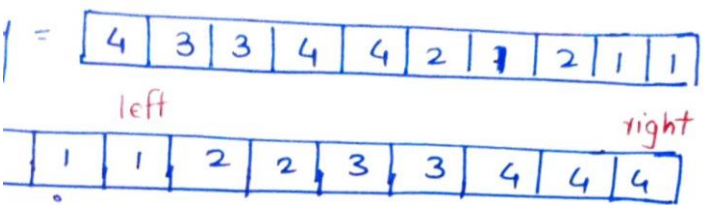
```
vect0Y          t    anS   st. be inC)Ð st-endC));
```

ans ;

0 Appyoach (Two pointets)

arr

soyt ihis

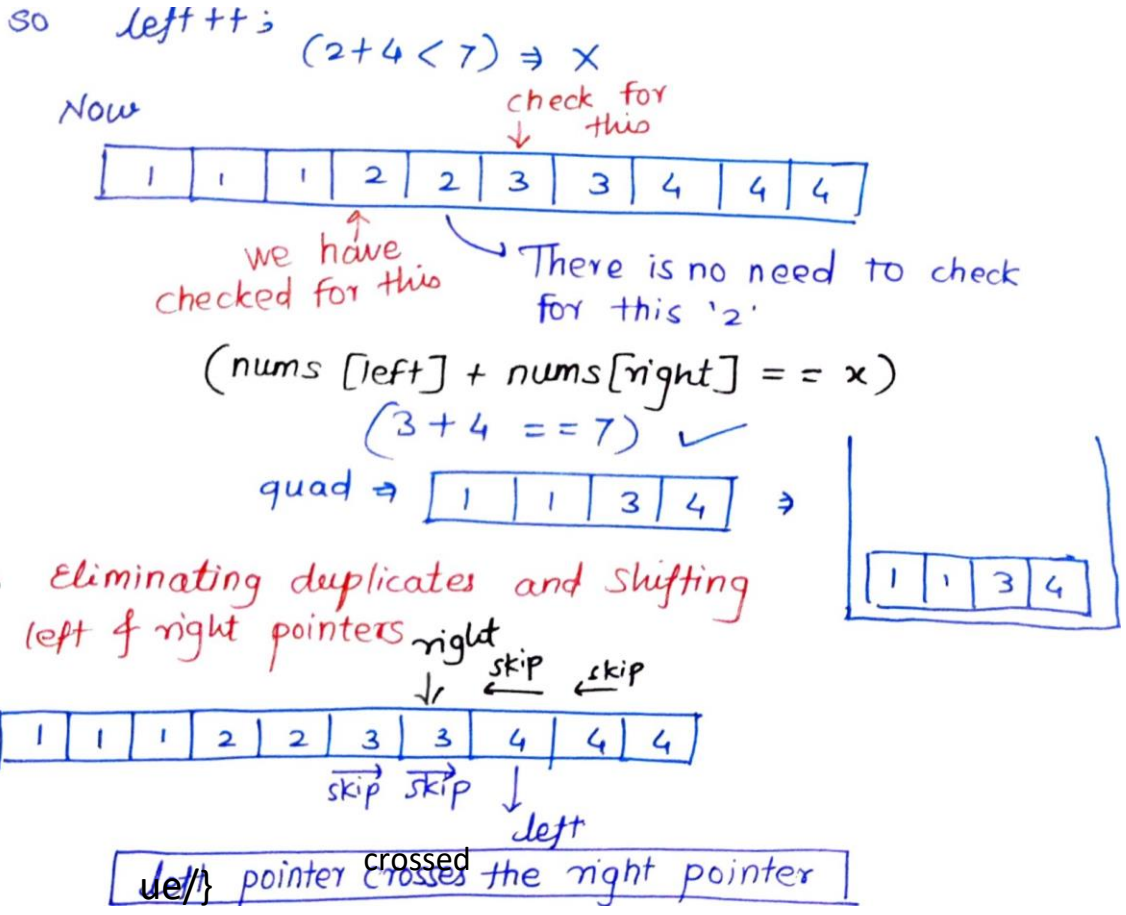


Target = 9

numsCóJ+ numsGJ = (+1 = 2 To find :-(7) → and n.ou->

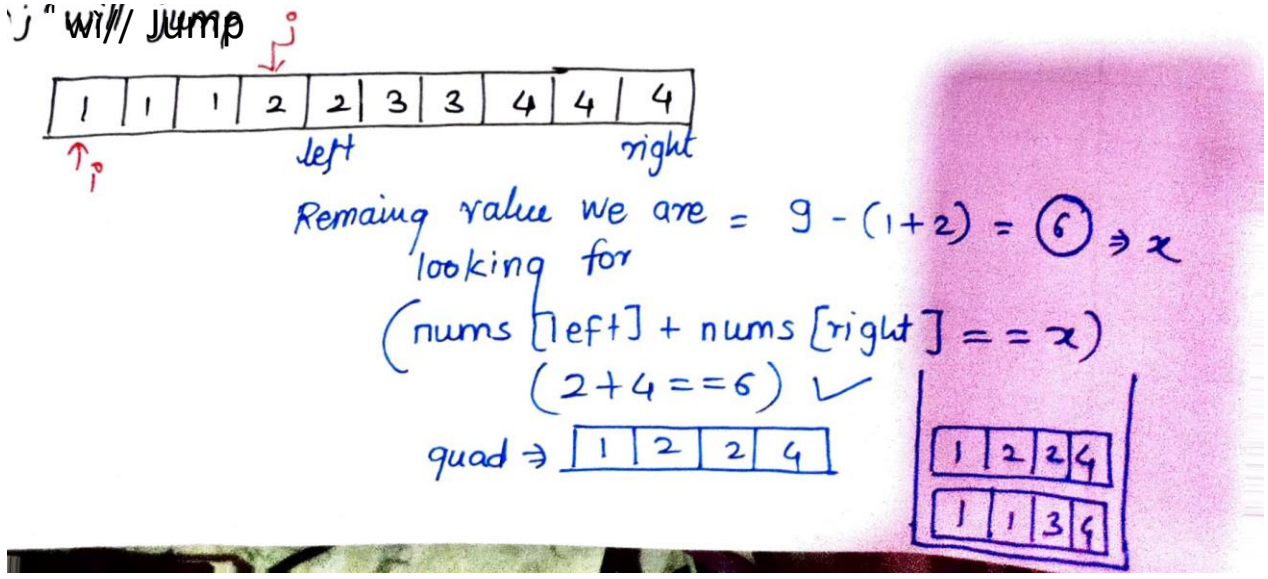
s[left]+ nurnsC-ri9htJ)

1 4 4 < 7) so no sol n



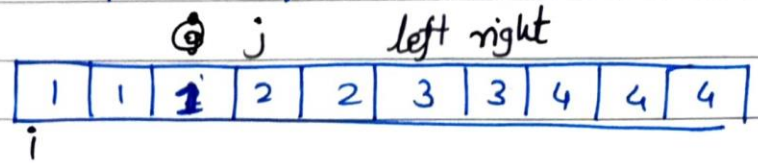
so +

Noup



RAISONI GROUP
— a vision beyond —

Now skip the duplicates



(3+3 == 6) ✓

quad | 1 | 2 | 3 | 3 | ⇒

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

likewise we can find

Time Complexity = $O(n^3)$
Space Complexity = $O(1)$

```
for (i = 0 to n)
   target_3 = target - nums[i]
    for (j = i+1 to n)
       target_2 = ~~target~~ (target_3) - nums[j]

       ~~front~~ = j+1        left = j+1 ;
       ~~back~~ =             right = n-1 ;
       While (left < right)
         if x = nums[left] + nums[right]
         if (x < target_2) ⇒ left++;
         else if (x > target_2) ⇒ right--;
         else {
              vector<int> quad(4,0)
              push all elements (i, j, left, right)
Avoiding duplicates
         while( left<right && nums[left] == quad[2]) ⇒ left++;
         while( left<right && nums[right] == quad[3]) ⇒ right--;
         }
       }
       while (j+1 < n && nums[j+1] == nums[j])
              j++;
       }
       while (i+1 < n && nums[i+1] == nums[i]) ⇒ i++;
```

of 'N'

21 congest Consecutive Sequence in an Arr

ou axe Siven an aintegef$.

you neeql to find +he Iengfh of fhe ta)hich contaíns uence conseceŒive

elements :-nums= 100 200 , [1, 2, 3, 4] output = 9

* Solutions

O Brute ôrce:-

SOI} +he int pyev =

```
for (i=1 to n)
    if (nums[i] = = prev + 1)
```

nu-ms Co] int ans z =

J ;

Cuy ++3 Check È Ju.Jf ìf au 41 nect -

Hu preuíou..oD else ìf (nums _ pyev)í

///Set

peu - nu.msCiJ ans =

max (ans, an) ; xetun•l

ans ;

Time Complexcity = O(nlogn) + O(n)
≈ O(nlogn)

Space complexity = O(1)

② Optimal Approach :- (using Hashset)

Insert all elements into HashSet;
Now traverse the ~~hashset~~ nums, and will check the (num - 1) element is present or not
then we'll do currentNum = num;
currentStreak = 1

then will increase the currentStreak until the (currentNum + 1) element is present in the HashSet.

longestStreak = max(longestStreat, CurrentStreak);

```
set<iut> hashSet;
for(int num : nums)
    hashSet.insert(num);

longestStreak = 0;
for(int num : ~~hashSet~~nums) {
    if(!hashSet.count(num-1)) {
        currentNum = num;
        currentStreak = 1;
        while(hashSet.count(currentNum + 1)) {
            currentNum += 1;
            currentStreak += 1;
        }
    longestStreak = max(longestStreak, currentStreak);
    }
}
return longestStreak;
```

T.C = O(N)
S.C = O(N)

Given an both posi+ive and negœfive integeys ,wc hœve +0 find the lehg+h of {he longea^{y} subaw the sumelements etkal tv zero .

:-nums= 1 a -

output = 5

subarr

u.)i+h sum o

[-3/3] Len z 2

L-l,6t-g1 (en s3

[-3, 3 , -t ,

6, -s) ⇒• Solu±ions

i CO ; -y + 'l) surn = o

; j<n ; j++)

foy 94m + = nuns C" 3 ; if (Sum= =0)

max(maxlen . j-i+1),

-yQh-wn

zero sum

Naive
Appyoach

5nìÐa,li' a vañable max,len : 0 Stoïed St-LbQ-ry -the with haxirneu.aL (e+l,o

(2) Traveyse He array from stQNt cud in;+ìalise a va+ab(e Sum —o a.)hìch sÌèye.g sum / ef subaxrw.Í sq-a-ins wit-h cumren+lndex

@ Tyawerse from nex+ el-uneat cumreül.ndex u.p fo n

Sum+ -z numsCiJ

if (sum co)

moK)en max(maxlen, j-i +1 .

-yetuyn rY)0$ten .

T.C = O(N²)
S.C = O(1)

RAISONI GROUP
a vision beyond

② Optimized Approach :-

Maintain (maxi) to store the maxlen of subarray

	0	1	2	3	4	5	6	7	8	9
nums =	1	-1	3	2	-2	-8	1	7	10	23

HashMap

(key, Value)

Now linearly traverse through the array, and we'll store the prefix sum

At i=0

sum = 1 → Add to hashmap with its index i.e. (1,0)

(1,0)

At i = 1

sum = 0

1	-1	⇒ This subarray with sum = 0

maxi = 2

At i = 2

sum = 3 → Add to hashmap (3,2)
(but before check if "3" exists in the hashmap or not)
(if it does not exist then add.

(5,3)
(3,2)
(1,0)

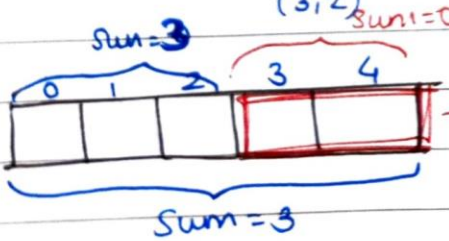
At i = 3

sum = 5 → Add to hashmap (5,3)

At i=4

sum = 5 - 2 = 3 (It is exists in the hashmap)

So this means that till index ⇒ 2 we had sum = 3 (3,2) sum1=0 and till 4 we have sum = 3



⟹ So this is one of the subarray.

its length = the index at which we have got S=3 − the index at which we have got S=3 (previously)

= 4 - 2

= (2)

Now At i=5

At i=6

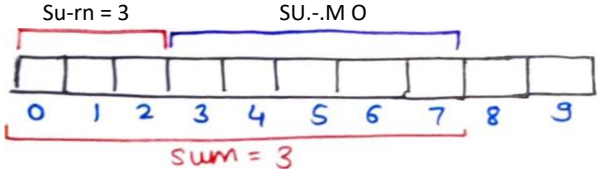
ûm1 = a -8 check -5 is pre..ge.»å in thL ho hm 07 not
ì f not i+ alon t-Dith i O

(-4,6)
(-5,5)
(5,3)
(3,2)
(1,0)

Su-m= -
= — 1.4

At i=7

sum = -4+7
= 3 ⟹



-Len = 7 - 2

At i=8

At i=9

maxi =
~~len~~ = max (len, maxi)
= max(5, 2)

maxi = 5

+01

Sum = 3 + 10
= 13 → Add (13, 8) in hashmap

(36, 9)
(13, 8)
(-4, 6)
(-5, 5)
(5, 3)
(3, 2)
(1, 0)
hashMap

Sum = 13 + 23 → Add (36, 9)
= 36

So the maximum length = 5

Add C) in h auh
m

= 36

So HLL

```
unordered_map<int, int> mp;
int maxi = 0
int sum = 0
for(int i=0; i<n; i++){
        sum += nums[i];
        if(sum == 0){
            maxi = i+1;
        }
        else{
            if(mp.find(sum) != mp.end()){
                maxi = max(maxi, i - mp[sum]);
            }
            else{
                mp[sum] = i;
            }
        }
}
return maxi
```

Time Complexity = ~~O(nlogn)~~ $O(n) + O(n\log n) \approx O(n\log n)$

Space Complexity = $O(n)$

6) -Count the numbey 0f S jiven XOR k

Given an ar 5f integerrs and e an integey ra, find ihe -frbtal numberr Of Sub TT hawìn b;twìse XOR dr au eleme.A.åJ

A = [4, 2, 2, 6, 4]

o/p = 4

Subarrays having XOR == B

[4,2]

[4,2,2,6,4]

Solutions ⇒

@ Brute force:-

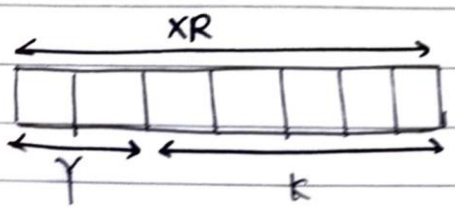
```
Jong          bng
filCiui 1_ J Î <
n J i + + ) 3
  for (iut j= i ; j<n ; j++)
      CAYT-XOX          A ci J;
        1     -xor ==

          3
    3

       coceu-t
```

Tme Omplexifi/ = Ô(nz)

CP1ce ComplexitÌ' = 0 (t)

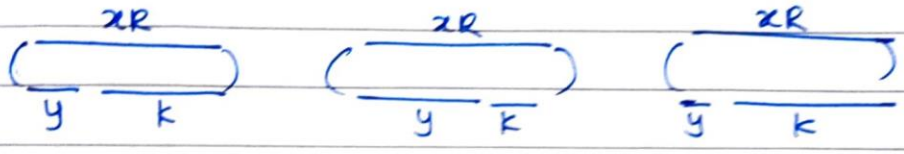
RAISONI GROUP
— a vision beyond —

② Optimized Approach :-



~~then~~ $Y \wedge k = XR$

$\Rightarrow$ ~~k~~ $\boxed{Y = XR \wedge k}$



there are multiple 'y's so

the $\boxed{\text{no. of } y\text{'s} = \text{no. of Subarrays}}$

dry Run :-

	0	1	2	3	4
nums =	4	2	2	6	4

$K = 6$

Assign XOR = 0, count = 0;

At i=0

XOR = 0 ∧ 4 = 4 → Does this giving k=6 (x)

Add to hashmap with its count

(4,1) → (6,1) (4,1)

Hashmap (prefix_XOR, count)

At i=1

XOR = 4 ∧ 2

= (6) ⇒ Yes it is giving

4, 2, 2, 6, 4

increment count ⇒ count++;

if it is present then we will increment the count

else Add (6,1) to hashmp.

$y = XOR \wedge k$

$= 6 \wedge k$

$\boxed{y = 0}$ → Check if '0' prefix_XOR is present in the Hashmap or not.

At i=2

xog = 6 ↄ

= 4 → y = xR ^ k

= 4 ^ 6

y = 2 → check in the hashmap

~~Add(4,1)~~ Add(4, cut)

But '4' is present already

So we'll just increment the cut

Cou-.&d

(6,1)

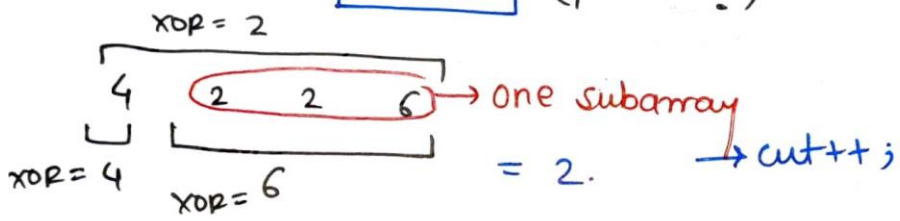
(4,2)

XOR = 4 ^ 6

= 2 → y = XOR ^ k

= 2 ^ 6

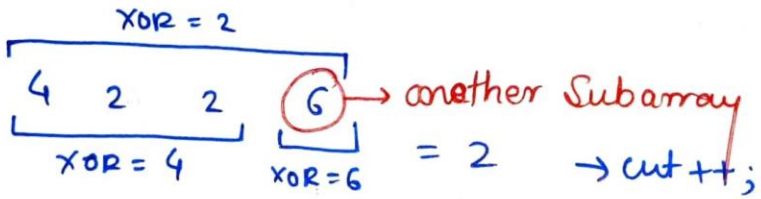
y = 4 (present!)



(2,1)

(6,1)

(4,3)



Count = 3

xog

Yopc

XOR = 2 ^ 4

= 6 → y = XOR ^ 6

cntt++;

cnt = 4

= 6 ^ 6

y = 0

Count = 4

T.C = O(nlogn)

S.C = O(n)

```
cnt=0
XOR=0
for (auto i: A) {
    XOR = XOR ^ i;
    if(XOR == B){
        cnt++;
    }
    if (freq.find (XOR ^ B) != freq.end()){
        cnt += freq[XOR ^ B];
    }
freq[XOR] += 1;
}

return cnt;
```

Gìven a sfrìng fìnd fhe lengh fhe Jon S(LbShin (a)iYhouf any yepeœtinj chamactem.

Ex :-s: qbca bcbb 't

So)uHons

(i) Brute force :-

Toke ttDO loops ⇒

Ô One -fraueMsiÍB The Stiing

(2) Anofhu nes+ed loop foy findinj differen+ subs+ri n we gs an

Will check -foy QIL subslYin one by one check foy eacb

evey element.

1f „the element is not *hem (De store -the elemeu-e haahså 0+he7Wìse break in from -the t^oop

~~int ans = 0;~~

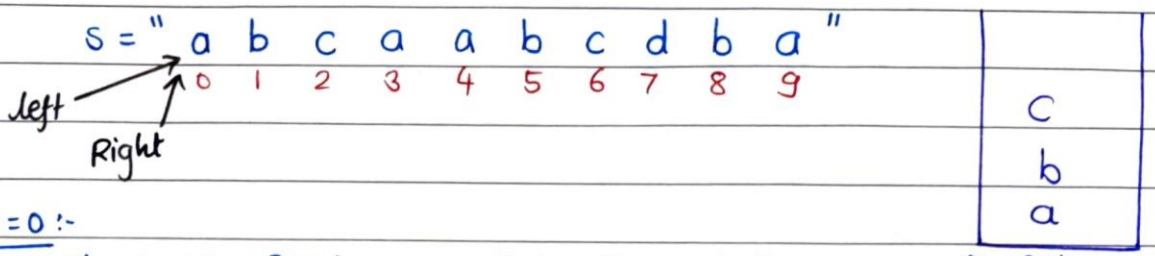
```
int count = INT_MIN;
for(int i=0; i<n; i++){
    unordered_set<int> st;
    for(int j=i; j<n; j++){
        if(st.find(s[j]) != st.end()){
            count = max(count, j-i);
            break;
        }
        st.insert(s[j]);
    }
}
return ~~ans~~ count;
```

Time Complexity = $O(n^2)$
Space Complexity = $O(n)$

RAISONI GROUP
— a vision beyond —

② Optimized Approach :-

len = 0



At i=0 :-

Check if s[i] is present in the set (Not present)

so Range (L - R) ⇒ This substring has no repeating characters

len = (r-l+1) = 0-0+1 = 1

update len = 1

insert the character into set

move the right pointer

At i=1 :-

'b' is not present

(L-R) ⇒ (r-l+1)

= 1-0+1 = 2

update ⇒ len = 2 (Insert b)

At i=2

'c' is not present in set

(L-R) ⇒ (r-l+1)

= (2-0+1)

= 3

update ⇒ len = 3 (Insert c)

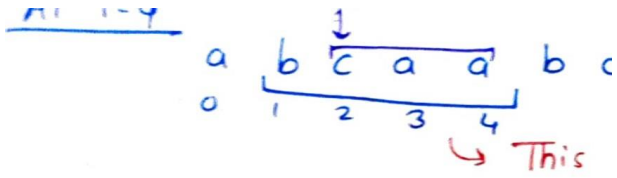
At i=3

'a' is present so we can say that (L-R) range has some repeating characters, so we've to remove that

remove 'a' from set & do `left++;`

(L-R) ✓ (r-l+1)

= [(3-1)+1] = 2+1 = 3 (len = 3) push 'a' to set



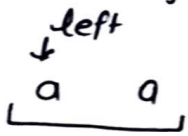
ha.) repeating chara cten

remove ⇒ s[left] ⇒ remove SHI) han e a-fin characterra

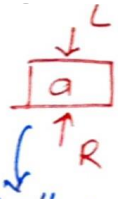
then left++;

Now c a a → It

remove c ⇒ left++;



—remove



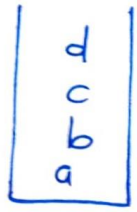
(It does not have repeating characters)

len = 1

But we'll not update the length

push this "a" to set

move the "R++;"



'b' is not present

len = 2 (No updation)

push b into set

'c' is not present

len = 3 (No updation)

push 'c' into set

'd' is not present

~~len = 4~~ (L - R) ⇒ (7 - 4) + 1

= 4

update len = 4

push 'd' into set

Now At i = 8

a b c d b

~~left++;~~

remove [left] ⇒ remove 'a'

left++;

b c d b

we still have 'b'

remove 'b'

left++;

c d b

len = 3 (No updation)

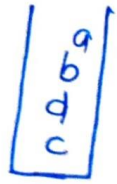
Now At i=9

'a' is not present

len = 4 ⇒ (L - R)

= (9 - 6) + 1

= 3 + 1 = 4



doe have -repea.hns

chqractem) ardœte

Time Complexity = O(2*n)
Space Complexity = O(n)

```
int maxlen = INT_MIN;
unordered_set <int> st;
int l=0
for( int r=0 ; r<n ; r++)
 {
   if (st.find(s[r]) != st.end())
    {
      while ( l <r && st.find (s[r]) != st.end())
        {
          st.erase (st [l]);
          l++;
        }
    }
   st. insert (s[r]);
  maxlen = max (maxlen , r-l+1);
}

return maxlen;
```

L IMe maf

a b C O Q b c

8 3

(c,2)
(b,1)
(a,0)

① puzb 'a' ⇒Cato)

@ pub s b' ⇒ (b,

@ 1010b i c t ⇒ Cc,2)
Glen=
is pres

Notø 'a' is in The mOf 1 [1] knoo 0' index '

at
' to (0+1)
⇓
1st index

So We 'will shifd º L' (Now update

(c,2)
(b,1)
(a,3)

-he ìndex a-fa)

a Q So MJil) 2 3 4

l = 3
a^{l}
↑r

(CIO

γ rq-sJ .

Catk)

RAISONI GROUP
a vision beyond

a, b, c

(c,6)
(b,5)
(a,4)

Now at i=7

'd' is there (4-7)

length = 7 - 4 + 1

= 4

(d,7)
(c,6)
(b,5)
(a,4)

Now at i=8 — jump

a b c d b

4 5 6 7 8

l = 5+1 = 6

c d b

(d,7)
(c,6)
(b,8)
(a,4)

Now at i=9

l r

c d b a

6 7 8 9

(6-9) ⇒ length = 4

(d,7)
(c,6)
(b,8)
(a,9)

Time Complexity = O(n)

Space Complexity = O(1) → Avg. case (S.C)

q ¯ string awe 256 chaM

vector d , ,intlefY=O Jright =

int den -o

(Dhile (n){

if (mp[s[right]] != -1) {

left = max (mp[s[right]] + 1 , left);

[s[right]]

Jen max (lent left+1) ;

right++ ;

'return len ;