Array

21 December 2022 18:03

que 1 :-

Union of Two Sorted Arrays

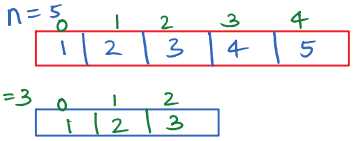
Easy Accuracy: 31.39% Submissions: 98K+ Points: 2

Union of two arrays can be defined as the common and distinct elements in the two arrays.
Given two sorted arrays of size **n** and **m** respectively, find their union.

Example 1:

Input:
n = 5, arr1[] = {1, 2, 3, 4, 5}
m = 3, arr2 [] = {1, 2, 3}
Output: 1 2 3 4 5
Explanation: Distinct elements including both the arrays are: 1 2 3 4 5.

Screen clipping taken: 21-12-2022 21:02



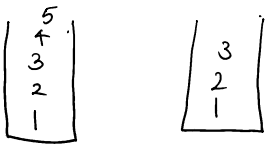
Bruteforce Approach :-
taking a HashSet And enter all the value in the hashset then put both.

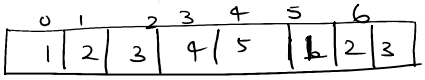
```
// n, m: size of arrays
class Solution
{
    //Function to return a list containing the union of the two arrays.
    public static ArrayList<Integer> findUnion(int arr1[], int arr2[], int n, int m)
    {
        // add your code here
            TreeSet <Integer> s=new TreeSet<>();  //Treeset is used for sorted
            ArrayList < Integer > Union=new ArrayList<>();
            for (int i = 0; i < n; i++)
              s.add(arr1[i]);
            for (int i = 0; i < m; i++)
              s.add(arr2[i]);
            for (int it: s)
              Union.add(it);

            return Union;
    }
}
```

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

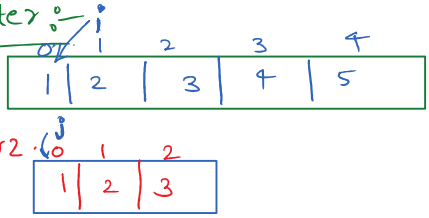
HSL

Screen clipping taken: 21-12-2022 18:20





Better Approach Using two pointer :-



Case I
While arr1[i] == arr2[j]

Case II arr1[i] > arr2[j]

Case III arr1[i] < arr2[j]

```
// n, m: size of arrays
class Solution
{
    //Function to return a list containing the union of the two arrays.
            public static ArrayList<Integer> findUnion(int arr1[], int arr2[], int n, int m)
            {
                // add your code here
                int i = 0, j = 0; // pointers
            ArrayList<Integer > Union=new ArrayList<>(); // Unionn vector
            while (i < n && j < m) {
              if (arr1[i] <= arr2[j]) // Case 1 and 2
              {
                if (Union.size() == 0 || Union.get(Union.size()-1) != arr1[i])
                  Union.add(arr1[i]);
                i++;
              } else // case 3
              {
                if (Union.size() == 0 || Union.get(Union.size()-1) != arr2[j])
                  Union.add(arr2[j]);
                j++;
              }
            }
            while (i < n) // IF any element left in arr1
            {
              if (Union.get(Union.size()-1) != arr1[i])
                Union.add(arr1[i]);
              i++;
            }
            while (j < m) // If any elements left in arr2
            {
              if (Union.get(Union.size()-1) != arr2[j])
                Union.add(arr2[j]);
              j++;
            }
            return Union;
    }
}
```

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

Screen clipping taken: 21-12-2022 21:25

Screen clipping taken: 21-12-2022 21:42

Intersection of two sorted arrays

Problem Statement: Find the intersection of two sorted arrays. OR in other words, Given 2 sorted arrays, find all the elements which occur in both the arrays.

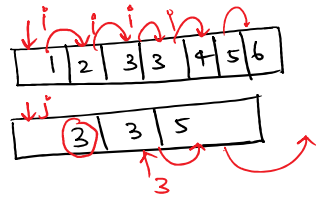
Examples:

Example 1:
Input:
A: [1 2 3 3 4 5 6]
, B: [3 3 5]
Output: 3,3,5
Explanation: We are given two arrays A and B.
The elements present in both the arrays
are 3,3 and 5.

Screen clipping taken: 21-12-2022 21:27

```java
class Solution {
    public int[] intersection(int[] nums1, int[] nums2) {
        HashSet<Integer>li = new HashSet<>();
        Arrays.sort(nums1);
        Arrays.sort(nums2);
        int i=0;
        int j=0;
        while(i<nums1.length && j<nums2.length){
                if(nums1[i] <nums2[j]){
                    i++;
                }
                else if(nums2[j] < nums1[i]){
                    j++;
                }
                else{
                    li.add(nums1[i]);
                        i++;
                        j++;

                }
        }
        int [] ans = new int [li.size()];
        int k = 0;
        for(int num: li) {
            ans[k++] = num;
        }

        return ans;
    }
}
```

Optimise Approach:-



1<3 → i++
2<3 → i++
3==3 → store 3 i++ j++
3==3 store 3 i++ j++
4<5 → i++
5==5 → i++ j++

TC:- O(n)
SC:- O(1)

A

Bruteforce Approach:-

```java
public static void main(String args[])
{
  int A[]= {1,2,3,3,4,5,6,7};
  int B[]= {3,3,4,4,5,8};

  ArrayList<Integer> ans=new ArrayList<>();
  int visited[]=new int[B.length]; // to maintain visited
  int i = 0, j = 0;
  for (i = 0; i < A.length; i++) {
    for (j = 0; j < B.length; j++) {

      if (A[i] == B[j] && visited[j] == 0) {
     //if element matches and has not been matched with any other before
        ans.add(B[j]);
        visited[j] = 1;

        break;
      } else if (B[j] > A[i]) break;
       //because array is sorted , element will not be beyond this
    }
  }
  System.out.print("The elements are: ");
```

TC:- O(n²)
SC:- O(n)

Screen clipping taken: 21-12-2022 21:44

Missing number

Basic Accuracy: 45.15% Submissions: 36K+ Points: 1

Bag Offers from Top Product Companies. Explore Exclusive Problems Now!

Vaibhav likes to play with numbers and he has **N** numbers. One day he was placing the numbers on the playing board just to count that how many numbers he has. He was placing the numbers in increasing order i.e. from **1** to **N**. But when he was putting the numbers back into his bag, some numbers fell down onto the floor. He picked up all the numbers but one number, he couldn't find. Now he has to go somewhere urgently, so he asks you to find the missing number.
NOTE: Don't use Sorting

first Approach:-

If there is only one missing no. then we can check this using:

sum → Original sum − sum.

Original sum = $\frac{4(4+1)}{2} = 10$

10 − 8 = 2

somewhere urgently, so he asks you to find the missing number.
NOTE: Don't use Sorting

Example 1:

Input:
N = 4
A[] = {1, 4, 3}
Output:
2
Explanation:

Screen clipping taken: 21-12-2022 21:55

Original sum = $\frac{4(4+1)}{2} = 10$

Sum = 8

10 − 8 = 2

```
25
26 int missingNumber(int arr[], int n)
27 {
28     // Your code goes here
29     int s = n*(n+1)/2;
30     int sum =0;
31     for(int i=0;i<n-1;i++){
32         sum += arr[i];
33     }
34     return s-sum;
35 }
```

Screen clipping taken: 21-12-2022 21:57

Better Approch!

1.XOR

```
public int missingNumber(int[] nums) { //xor
    int res = nums.length;
    for(int i=0; i<nums.length; i++){
        res ^= i;
        res ^= nums[i];
    }
    return res;
}
```

Screen clipping taken: 21-12-2022 22:03

1	4	3

Que 2 -

485. Max Consecutive Ones

Hint

Easy 3.7K 417

Companies

Given a binary array `nums`, return *the maximum number of consecutive* `1`*'s in the array.*

Example 1:

Input: nums = [1,1,0,1,1,1]
Output: 3
Explanation: The first two digits or the last three digits are consecutive 1s. The maximum number of consecutive 1s is 3.

Example 2:

```
1  class Solution {
2      public int findMaxConsecutiveOnes(int[] nums) {
3          int cnt =0;
4          int maxi = 0;
5          for(int i=0;i<nums.length;i++){
6              if(nums[i] == 1){
7                  cnt++;
8              }
9              else{
10                 cnt =0;
11             }
12             maxi = Math.max(maxi, cnt);
13         }
14         return maxi;
15     }
16 }
```

1	1	0	1	1	1

cnt = 0 1 2 0 1 2 3

maxi = 0 1 2 3

return maxi

Screen clipping taken: 21-12-2022 22:20

Que 3 -

Longest Sub-Array with Sum K

Medium Accuracy: 24.64% Submissions: 106K+ Points: 4

Bag Offers from Top Product Companies. Explore Exclusive Problems Now!

Given an array containing **N** integers and an integer **K**., Your task is to find the length of the longest Sub-Array with the sum of the elements equal to the given value **K**.

Example 1:

Input :
A[] = {10, 5, 2, 7, 1, 9}
K = 15
Output : 4
Explanation:
The sub-array is **{5, 2, 7, 1}**.

Screen clipping taken: 21-12-2022 22:23

Bruteforce.

k = 15

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

```
for (i = 0 → n)
    j = i → n.
    sum += arr[j]
    if (sum == k)
maxLen. = max(maxLen, j-i+1)
```

O(n2) TC :-
SC O(1)

(I) Sum = 0 i = 0 j = 0 i = 1 j = 1

```
48
49 class Solution{
50
```

(1) Sum = 0 i = 0 j = 0 sum = 10 i = 1 j = 1

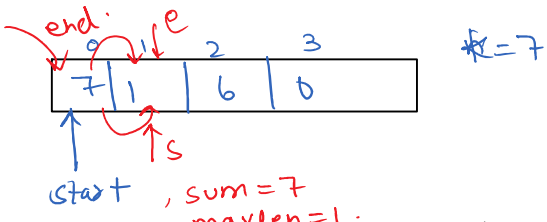
O(n2) TC :-
SC O(1)

```
class Solution{

    // Function for finding maximum and value pair
    public static int lenOfLongSubarr (int arr[], int n, int k) {
        //Complete the function
        int sum =0;
        int maxlen =0;
        for(int i=0;i<n;i++){
            for(int j=0;j<n;j++){
                sum += arr[j];
                if(sum ==k){
                    maxlen = Math.max(maxlen , (j-i +1) );
                }
            }
        }
        return maxlen;
    }


}
```

Screen clipping taken: 21-12-2022 23:02

Efficient solution :-

maxlen = 0.

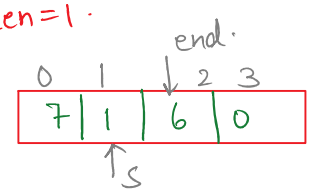


k = 7

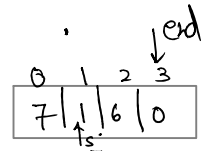
, sum = 7
maxlen = 1.

Adding 1 → 7+1 = 8.
start+1
end = 1
sum = 1
len = 1

curr sum = 7
maxlen = max (1, maxlen).
= 1



sum = 7
maxlen = 2



sum = 7
maxlen = 3

Screen clipping taken: 21-12-2022 23:12

```
class Solution{

    // Function for finding maximum and value pair
    public static int lenOfLongSubarr (int A[], int N, int K) {
        //Complete the function
                int i = 0, j = 0, sum = 0;
            int maxLen = Integer.MIN_VALUE;

            while (j < N) {
              sum += A[j];
              if (sum < K) {
                j++;
              }
              else if (sum == K) {
                maxLen = Math.max(maxLen, j-i+1);
                j++;
              }
              else if (sum > K) {
                while (sum > K) {
                  sum -= A[i];
                  i++;
                }
                if(sum == K){
                  maxLen = Math.max(maxLen, j-i+1);
                }
                j++;
              }
            }
            return maxLen;
    }
```

→ agar sum se bada ho gaya then start ko increase.

o

Problem | Editorial | Submissions | Comments

Medium Accuracy: 53.69% Submissions: 47K+ Points: 4

Bag Offers from Top Product Companies. Explore Exclusive Problems Now!

Given a sorted array **A[]** of **N** positive integers having all the numbers occurring exactly twice, except for one number which will occur only once. Find the number occurring only once.

Example 1:

Input:
N = 5
A = {1, 1, 2, 5, 5}
Output: 2
Explanation:
Since 2 occurs once, while other numbers occur twice,

Java (1.8) Average Time: 20m Start Timer

```
// } Driver Code Ends


//User function Template for Java

class Sol
{
    public static int search(int A[], int N)
    {
        // your code here
        int ans = 0;
        for(int i=0;i<N;i++){
            ans = ans ^ A[i];
        }
        return ans;
    }
}
```

1^1 = 0
5^5 = 0
0^2 = 2

TC = O(n)
SC = O(1)

Problem | Editorial | Submissions | Comments

Medium Accuracy: 53.69% Submissions: 47K+ Points: 4

Bag Offers from Top Product Companies. Explore Exclusive Problems Now!

Given a sorted array **A[]** of **N** positive integers having all the numbers occurring exactly twice, except for one number which will occur only once. Find the number occurring only once.

Example 1:

Input:
N = 5
A = {1, 1, 2, 5, 5}
Output: 2
Explanation:
Since 2 occurs once, while other numbers occur twice, 2 is the answer.

Java (1.8) Average Time: 20m Start Timer

```
// } Driver Code Ends


//User function Template for Java

class Sol
{
    public static int search(int A[], int N)
    {
        // your code here
        int ans = 0;
        for(int i=0;i<N;i++){
            ans = ans ^ A[i];
        }
        return ans;
    }
}
```

1^1 = 0
5^5 = 0
0^2 = 2

TC = O(n)
SC = O(1)

Screen clipping taken: 21-12-2022 23:17

que:- Search in a Matrix

Screen clipping taken: 21-12-2022 23:31

Search in a matrix

Easy Accuracy: 41.62% Submissions: 87K+ Points: 2

Bag Offers from Top Product Companies. Explore Exclusive Problems Now!

Given a matrix **mat**[][] of size **N** x **M**, where every row and column is sorted in increasing order, and a number **X** is given. The task is to find whether element **X** is present in the matrix or not.

Example 1:

Input:
N = 3, M = 3
mat[][] = 3 30 38
44 52 54
57 60 69
X = 62
Output:
0
Explanation:
62 is not present in the

first approach is Brute force.
TC :- O(n^2).
SC :- O(n)

Second Approach :-
Using Two pointer

```
[ 3  30  38 ]  0
[ 44 52  54 ]
[ 57 60  69 ]  N-1
```

for(i=0 → n-1)

Binary search Approach

```
si=0
[ 3  30  38 ]
[ 44 52  54 ]
[ 57 60  69 ]  N-1
          ei
```

int ridx = -1
if [arr[mid][m-1] == x)
{ true;
}

while(si <= ei)

>x:
ridx = mid
ei = mid - 1

<x:
si = mid+1

concept

```
public static int matSearch(int mat[][], int n, int m, int x)
{
    // your code here
    int si = 0;
    int ei = n-1;
    int ridx = -1;
    while(si <= ei){
        int mid  = (si  + ei)/2;
        if(mat[mid][m-1] == x){
            return 1;
        }
        else if(mat[mid][m-1] > x){
            ridx = mid;
            ei = mid -1;
        }
        else{
            si = mid +1;
        }
    }
    if(ridx == -1){
        return 0;
    }
    si =0;
    ei = m-1;
    while(si <= ei){
        int mid = (si + ei)/2;
        if(mat[ridx][mid] == x){
            return 1;
        }
        else if(mat[ridx][mid] > x){
            ei = mid -1;
        }
        else{
            si = mid +1;
        }
    }
    return 0;
}
```

if(ridx == -1)
return false.

si=0
ei = m-1

if (mat[ridx][mid] == x.
return true.

>x:
ei = mid-1

<x:
si = mid+1

return false;

Screen clipping taken: 21-12-2022 23:56

que:-

Row with max 1s

Medium Accuracy: 33.09% Submissions: 163K+ Points: 4

Bag Offers from Top Product Companies. Explore Exclusive Problems Now!

Given a boolean 2D array of n x m dimensions where each row is sorted.
Find the 0-based index of the first row that has the maximum number of **1's**.

Example 1:

Input:
N = 4 , M = 4
Arr[][] = {{0, 1, 1, 1},
{0, 0, 1, 1},
{1, 1, 1, 1},
{0, 0, 0, 0}}
Output: 2
Explanation: Row 2 contains **4** 1's (0-based indexing).

max = 0 index = -1

```java
// } Driver Code Ends


//User function Template for Java

class Solution {
    int rowWithMax1s(int arr[][], int n, int m) {
        // code here
        int maxLifes = 0;
        int index = -1;

        for (int i = 0; i < n; i++) {
            int currLevelLifes = 0;
            for (int j = 0; j < m; j++) {
                if (arr[i][j] == 1) {
                    currLevelLifes++;
                }
            }
            if (currLevelLifes > maxLifes) {
                maxLifes = currLevelLifes;
                index = i;
            }
        }

        return index;
    }
}
```

que:-

Screen clipping taken: 22-12-2022 00:03

Two sum of Using BruteForce Approach

1. Two Sum

Hint

Easy 41.3K 1.3K

Companies

Given an array of integers `nums` and an integer `target`, return *indices of the two numbers such that they add up to* `target`.

You may assume that each input would have ***exactly* one solution**, and you may not use the *same* element twice.

You can return the answer in any order.

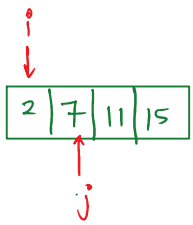
Example 1:

```
Input: nums = [2,7,11,15], target = 9
Output: [0,1]
```

```java
class Solution {
    public int[] twoSum(int[] nums, int target) {
        int n = nums.length;
        int ans[] = new int[2];
        for(int i=0;i<n-1;i++){
            for(int j = i+1;j<n;j++){
                if(nums[i] + nums[j] == target){
                    ans[0] =i;
                    ans[1] =j;
                    return ans;
                }
            }
        }
        return ans;
    }
}
```

Console

Screen clipping taken: 22-12-2022 14:07



int ans[] = new int[2] target = 9

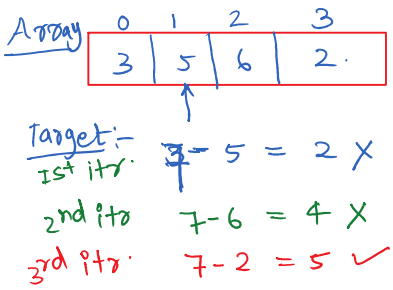
check is 2+7 == 9 → True.

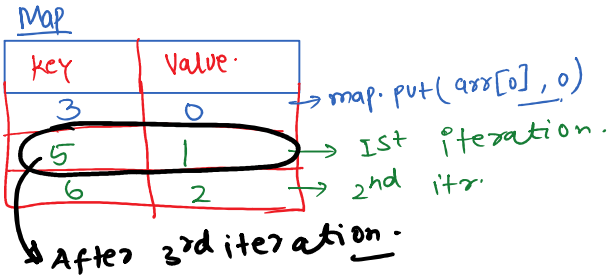
ans[0] = i
ans[1] = j
return ans;

return ans;

TC:- O(n²)
SC:- O(1)

Efficient approach :- Using HashMap :-






```java
class Solution {
    public int[] twoSum(int[] nums, int target) {
        int ans[] = new int[2];
```

TC:- O(n)

```java
class Solution {
    public int[] twoSum(int[] nums, int target) {
        int ans[] = new int[2];
        HashMap<Integer, Integer>mp = new HashMap<Integer, Integer>();
        mp.put(nums[0] , 0);
        for(int i=1;i<nums.length;i++){
                int val = target - nums[i];
                if(mp.containsKey(val)){
                    // System.out.println("checking "+mp.containsKey(val));
                    ans[0] = mp.get(val);
                    // System.out.println("the first val"+mp.get(val));
                    ans[1] = i;
                    //  System.out.println("the second val"+ans[1]);
                    return ans;
                }
                else{
                    mp.put(nums[i] , i);
                }
        }
        return ans;
    }
}
```

TC:- O(n)
SC:- O(1)

Screen clipping taken: 22-12-2022 14:52

que 2:-

Description | Discussion (31) | Solutions (7.8K) | Submissions

75. Sort Colors

Hint

Medium | 13.3K | 484

Companies

Given an array `nums` with `n` objects colored red, white, or blue, sort them **in-place** so that objects of the same color are adjacent, with the colors in the order red, white, and blue.

We will use the integers `0`, `1`, and `2` to represent the color red, white, and blue, respectively.

You must solve this problem without using the library's sort function.

Example 1:

```
Input: nums = [2,0,2,1,1,0]
Output: [0,0,1,1,2,2]
```

Screen clipping taken: 22-12-2022 14:56

Approach-I
Using Sorting-
TC:- O(nlogn)
SC:- O(1)

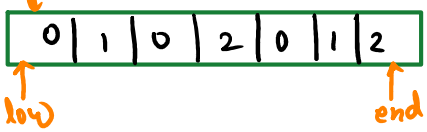
Approach-II
Using counting sort
cnt2=0, cnt1=0, cnt2=0;
O(n)+O(n)
SC:- O(1)

arr → 

if (arr[i] = 0) → cnt0++ → 2.
if (arr[i] == 1) → cnt1++ → 2
if (arr[i] == 2) → cnt2++ → 2.

i=0; while(cnt0>0) arr[i++] = 0 cnt0--
while(cnt1>0) arr[i++] = 1 cnt1--
while(cnt2>0) arr[i++] = 2 cnt2--

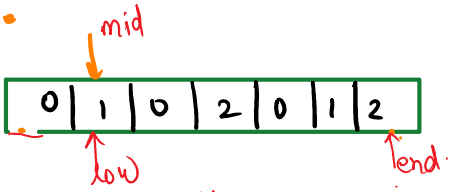
Approach-III :-
Dutch flag Algorithm :-



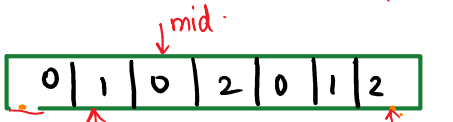
Ist itr a[l] == a[m] swap(a[l], a[m]) l++ m++

mid

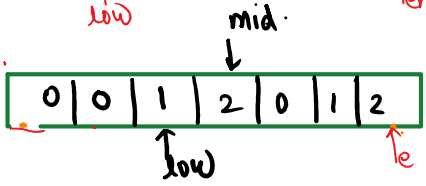
If a[mid] == 1 → no swapping.



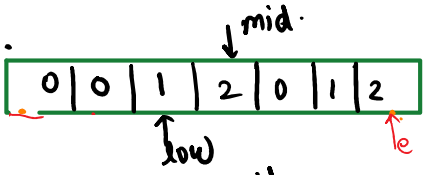
If a[mid] == 1 → no swapping.
mid++



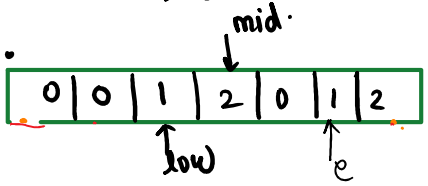
3rd itr
a[mid] == 0
swap (a[l], a[mid] l++ mid++.

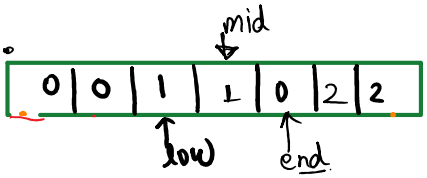


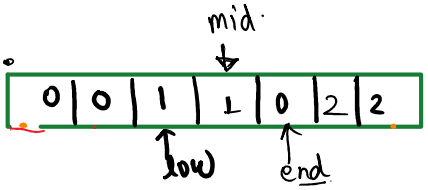
4th itr



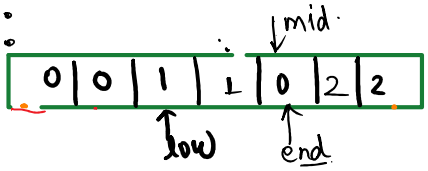
a[mid] == 2
swap (a[mid], a[high] high--



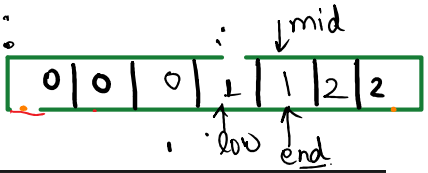




arr[mid] == 1 → mid++.



arr[mid] != 0
swap (arr[mid], arr[low])



mid <= end → condition reached

Screen clipping taken: 22-12-2022 15:39

```java
class Solution {
    public void sortColors(int[] arr) {
        int start = 0;
        int mid = 0;
        int end = arr.length -1;
        while(mid <= end){
            if(arr[mid] == 0){
                int temp = arr[start];
                arr[start] = arr[mid];
                arr[mid] = temp;
                start++;
                mid++;
            }

            else if(arr[mid] == 1){
                mid++;
```

```java
class Solution {
    public void sortColors(int[] arr) {
        int start = 0;
        int mid = 0;
        int end = arr.length -1;
        while(mid <= end){
            if(arr[mid] == 0){
                int temp = arr[start];
                arr[start] = arr[mid];
                arr[mid] = temp;
                start++;
                mid++;
            }

            else if(arr[mid] == 1){
                mid++;
            }
            else{
                int temp = arr[end];
                arr[end] = arr[mid];
                arr[mid] = temp;
                end--;
            }
        }
    }
}
```

que 3°

Majority Element

Medium Accuracy: 27.82% Submissions: 399K+ Points: 4

Bag Offers from Top Product Companies. Explore Exclusive Problems Now!

Given an array **A** of **N** elements. Find the majority element in the array. A majority element in an array A of size N is an **element that appears more than N/2 times in the array**.

Example 1:

Input:
N = 3
A[] = {1,2,3}
Output:
-1
Explanation:
Since, each element in
{1,2,3} appears only once so there
is no majority element.

Screen clipping taken: 22-12-2022 15:58

Example 2:

Input:
N = 5
A[] = {3,1,3,3,2}
Output:
3
Explanation:
Since, 3 is present more
than N/2 times, so it is
the majority element.

Screen clipping taken: 22-12-2022 16:05

Approach -I Using nested loop Approach.

majority = -1;

i=0

1	2	3

j=0.

Ist itr → if (arr[i] == arr[j]
majority ++

if (majority > n/2)

return. return. X[i];

else return -1;

this case majority = -1

Approach -2

sort → 1 2 3 3 3 N = 5 5/2 = 2

nums[2] = 3

TC:- O(nlogn)
SC:- O(1)

0	1	2	3	4
3	1	3	3	2

int cnt=0;
int maj =0;

Approach- 3°

Ist itr →. cnt=0.
majo = arr[i]
if (x[i] == majo)
cnt++;
else cnt--
return majo

TC:- O(n)
SC:- O(1)

Approach 4:-

0	1	2	3	4
3	1	3	3	2

int cnt=0
int Candidate =0.

Approach 4 :-

0	1	2	3	4
3	1	3	3	2

```
int cnt=0
int Candidate =0.
for (int num : nums)
  {. if (cnt==0)
       candidate=nums;
     if (num==candidate)
       count++
     else
       count--
```

Dry Run :-

Ist itr :- cnt==0 → True → cardi = 3.

cardi == num → T.

↳ cnt=1

2nd itr cnt==1 cnt-- → cnt = 0

3rd itr. cnt = 0 cardi = 3 → cnt = 1

4th itr. cnt = 1 cand =3 → cnt = 2.

5th itr cnt=2. cand = 3 → cnt=1

↳ return candidate :

que 4 :-

53. Maximum Subarray

Medium 27K 1.2K

Companies

Given an integer array `nums`, find the subarray which has the largest sum and return *its sum*.

Example 1:

```
Input: nums = [-2,1,-3,4,-1,2,1,-5,4]
Output: 6
Explanation: [4,-1,2,1] has the largest sum = 6.
```

Screen clipping taken: 22-12-2022 17:09

-2	-1	-3	4	-1	2	1	-5	4

maxi = 0

j j = i

1st itr.

subarray sum = 0

↳ += arr[j]

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

2nd itr → -1 -4 0 -1 1

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

maxi = 1

1st itr

Sum =-2	-1	-4	0	-1	1	2	-3	1

sum = 2.

maxi=0 maxi=1

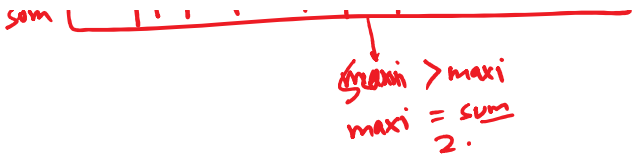
2nd itr.

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

sum	j	-2	2	1	3

O(n²) TC :-

O(1) SC :-



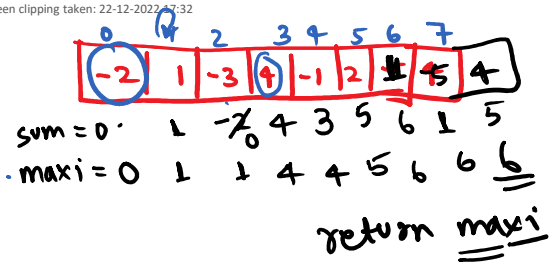
Better Approach:-

Using single loop:-

```
int sum = 0;
maxi = -∞
for (i = 0 → n)
    sum += arr[i]
    maxi = max(maxi, sum).
    if (sum < 0)
        sum = 0.
```

Input: nums = [-2,1,-3,4,-1,2,1,-5,4]
Output: 6
Explanation: [4,-1,2,1] has the largest sum = 6.

Screen clipping taken: 22-12-2022 17:32




```
class Solution {
public:
    int maxSubArray(vector<int>& nums) {
        int sum =0;
        int ans = INT_MIN;
        for(int i=0;i<nums.size();i++){
            sum = sum + nums[i];
            ans = max(ans ,sum);
            if(sum <0){
                sum = 0;}
        }
        return ans;
    }
};
```

Screen clipping taken: 22-12-2022 17:41

Screen clipping taken: 22-12-2022 19:22

que 5:-

Max sum in sub-arrays

Easy Accuracy: 43.26% Submissions: 16K+ Points: 2

Bag Offers from Top Product Companies. Explore Exclusive Problems Now!

Given an array, find maximum sum of smallest and second smallest elements chosen from all possible sub-arrays. More formally, if we write all (nC2) sub-arrays of array of size >=2 and find the sum of smallest and second smallest, then our answer will be maximum sum among them.

Example 1:

Input : arr[] = [4, 3, 1, 5, 6]
Output : 11
Subarrays with smallest and
second smallest are,
[4, 3]
smallest = 3 second smallest = 4
[4, 3, 1]
smallest = 1 second smallest = 3
[4, 3, 1, 5]

```java
class Solution {

    public static long pairWithMaxSum(long arr[], long n)
    {
        // Your code goes here
        long long max = INT_MIN;
        long long sum =0;
        for(long long i =0;i<n;i++){
            sum += arr[i];
            if(sum > max){
                max = sum;
            }
            if(sum < 0){
                sum =0;
            }
            if(i >=1){
                sum -= arr[i-1];
            }
        }
        return max;
    }
}
```

```
[4, 3, 1, 5]
smallest = 1    second smallest = 3
[4, 3, 1, 5, 6]
smallest = 1    second smallest = 3
[3, 1]
smallest = 1    second smallest = 3
[3, 1, 5]
smallest = 1    second smallest = 3
[3, 1, 5, 6]
smallest = 1    second smallest = 3
[1, 5]
smallest = 1    second smallest = 5
[1, 5, 6]
smallest = 1    second smallest = 5
[5, 6]
smallest = 5    second smallest = 6
Maximum sum among all
above choices is, 5 + 6 = 11
```

Screen clipping taken: 22-12-2022 17:43

Screen clipping taken: 22-12-2022 17:43

que :-

121. Best Time to Buy and Sell Stock

Easy 22K 695

Companies

Screen clipping taken: 22-12-2022 21:50

You are given an array `prices` where `prices[i]` is the price of a given stock on the i^{th} day.

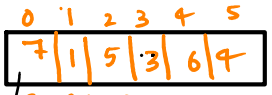
You want to maximize your profit by choosing a **single day** to buy one stock and choosing a **different day in the future** to sell that stock.

Return *the maximum profit you can achieve from this transaction*. If you cannot achieve any profit, return `0`.

Example 1:

```
Input: prices = [7,1,5,3,6,4]
Output: 5
Explanation: Buy on day 2 (price = 1) and sell on day 5 (price = 6), profit = 6-1 = 5.
Note that buying on day 2 and selling on day 1 is not allowed because you must buy before you sell.
```

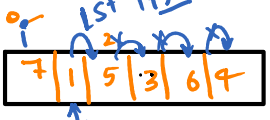
Brute force Approach.
Using two loop :-



maxprofit = 0.
for(i=0 → n)
j = i+1 → n.

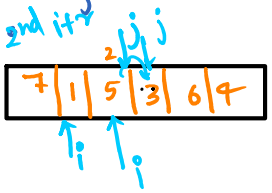
Screen clipping taken: 22-12-2022 22:01

Brute force Dry Run :-



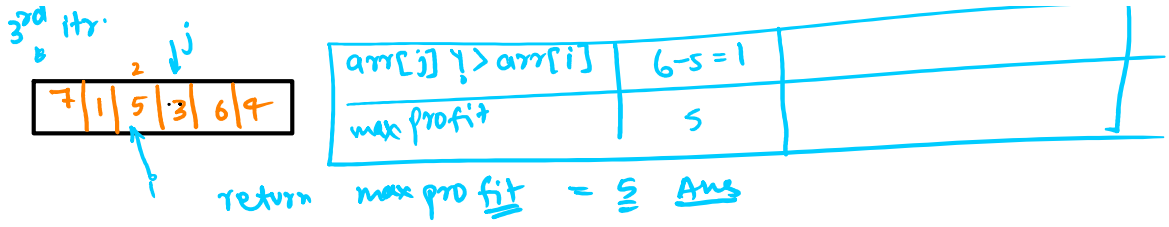
if(arr[j] > arr[i]
maxpro = max(arr[j]-arr[i], maxpro)
return maxpro;

① itr :- a[0]x



arr[j] > arr[i]	3 3-1 = 2	6 6-1	4 4-1
5-4 = 4			
maxpro(4,0) = 4	4	5	3

arr[j] !> arr[i]	6-5 = 1



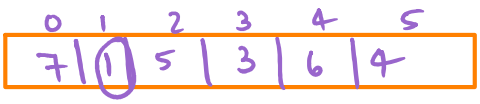
Screen clipping taken: 22-12-2022 22:13

```java
class Solution {
    public int maxProfit(int[] prices) {
        int maxpro =0;
        for(int i=0;i<prices.length;i++){
            for(int j = i+1;j<prices.length;j++){
                if(prices[j] > prices[i]){
                    maxpro = Math.max(prices[j] - prices[i] , maxpro);
                }
            }
        }
        return maxpro;
    }
}
```

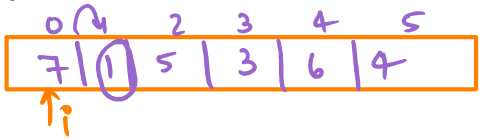
TC :- $O(n^2)$
SC :- $O(1)$

Efficient Approach :- Using kaden's algorithm.

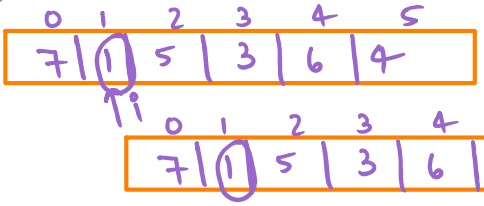
minprices = INT-MAX.
maxprofit = 0;
maxpro = max(prices[i] - min , maxprofit)



dry Run :-
1st itr :- .. maxpro = 0
minpric = +∞.



i = 0
minprices = min(7, ∞)
= 7
maxprofit = max(0-7, 0) = 0.

i = 1


min prices = min(7,1) = 1
maxprofit = max(1-1, 0)
= 0

↳ so on this give the Ans

```java
class Solution {
    public int maxProfit(int[] prices) {
        int maxpro =0;
        int minprices = Integer.MAX_VALUE;
        for(int i=0;i<prices.length;i++){
            minprices = Math.min(minprices, prices[i]);
            maxpro = Math.max(maxpro , prices[i] - minprices);
        }
        return maxpro;
    }
}
```

TC :- O(n)
SC :- O(1)

Screen clipping taken: 22-12-2022 22:26

Screen clipping taken: 22-12-2022 22:24

que :-

2149. Rearrange Array Elements by Sign

Medium 1.1K 64 Companies

You are given a **0-indexed** integer array `nums` of **even** length consisting of an **equal** number of positive and negative integers.

You should **rearrange** the elements of `nums` such that the modified array follows the given conditions:

Every **consecutive pair** of integers have **opposite signs**.

For all integers with the same sign, the **order** in which they were present in `nums` is **preserved**.

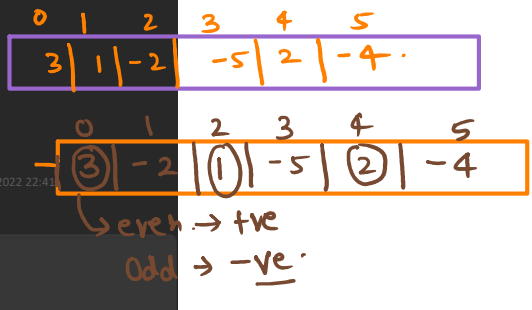
The rearranged array begins with a positive integer.

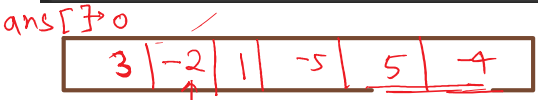
Return *the modified array after rearranging the elements to satisfy the aforementioned conditions.*

Screen clipping taken: 22-12-2022 22:41

Example 1:

```
Input: nums = [3,1,-2,-5,2,-4]
Output: [3,-2,1,-5,2,-4]
Explanation:
The positive integers in nums are [3,1,2]. The negative integers are [-2,-5,-4].
The only possible way to rearrange them such that they satisfy all conditions is [3,-2,1,-5,2,-4].
Other ways such as [1,-2,2,-5,3,-4], [3,1,2,-2,-5,-4], [-2,3,-5,1,-4,2] are incorrect because they do not satisfy one or more conditions.
```





even=0
odd=0

```
if(arr[i]>0)
    ans[even] = arr[i]
    even = even+2;
```

TC :- O(n)
SC :- O(n)

```java
class Solution {
    public int[] rearrangeArray(int[] nums) {
        int ans[] = new int[nums.length];
        int even =0;
        int odd =1;
        for(int i=0;i<nums.length;i++){
            if(nums[i] > 0){
                ans[even] = nums[i];
                even  =even + 2;
                }
            else{
                ans[odd] = nums[i];
                odd = odd +  2;
            }
        }
        return ans;
    }
}
```

que :- next permutations :-

31. Next Permutation

Medium 13.8K 3.9K Companies

A **permutation** of an array of integers is an arrangement of its members into a sequence or linear order.

- For example, for `arr = [1,2,3]`, the following are all the permutations of `arr`: `[1,2,3]`, `[1,3,2]`, `[2, 1, 3]`, `[2, 3, 1]`, `[3,1,2]`, `[3,2,1]`.

The **next permutation** of an array of integers is the next lexicographically greater permutation of its integer. More formally, if all the permutations of the array are sorted in one container according to their lexicographical order, then the **next permutation** of that array is the permutation that follows it in the sorted container. If such arrangement is not possible, the array must be rearranged as the lowest possible order (i.e., sorted in ascending order).

- For example, the next permutation of `arr = [1,2,3]` is `[1,3,2]`.
- Similarly, the next permutation of `arr = [2,3,1]` is `[3,1,2]`.
- While the next permutation of `arr = [3,2,1]` is `[1,2,3]` because `[3,2,1]` does not have a lexicographical larger rearrangement.
- While the next permutation of `arr = [3,2,1]` is `[1,2,3]` because `[3,2,1]` does not have a lexicographical larger rearrangement.

Given an array of integers `nums`, *find the next permutation of* `nums`.

The replacement must be **in place** and use only constant extra memory.

Brute force Approach is to generate All the permutation & then check it will take time complexity of .

Efficient Approach :-

rearrangement.

Given an array of integers `nums`, *find the next permutation of* `nums`.

The replacement must be **in place** and use only constant extra memory.

Example 1:

```
Input: nums = [1,2,3]
Output: [1,3,2]
```

Example 2:

```
Input: nums = [3,2,1]
Output: [1,2,3]
```

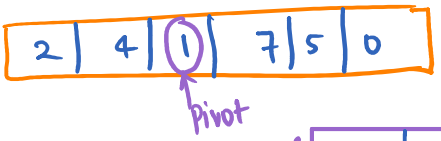
Efficient Approach.

[3, 1, 3] = next greater number is 331
[5, 1, 3] = next greater number is 531
[1, 2, 3] = next greater number is 132
[1, 3, 5, 4] = next greater number is 1435
[3, 2, 1] = we can't form a number greater than the current number from all the possible permutations

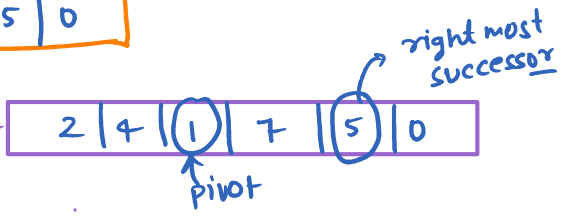
Initial sequence.

find longest non increasing suffix

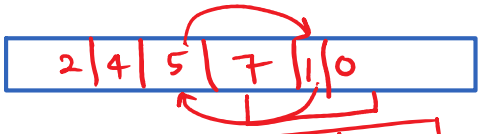
find pivot



find rightmost successor of the pivot in the suffix

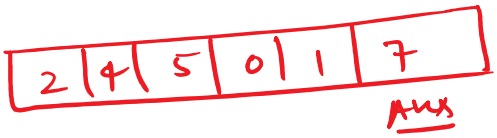


Swap the pivot & Right most successor.



Reverse the suffix

final sequence



Rule :-

find pivot :- arr[i] < arr[i+1] → when doesn't follow property non-increasing seq.

if pivot no exist then reverse all the array.

Otherwise iterate the array from end and find successor of pivot in suffix.

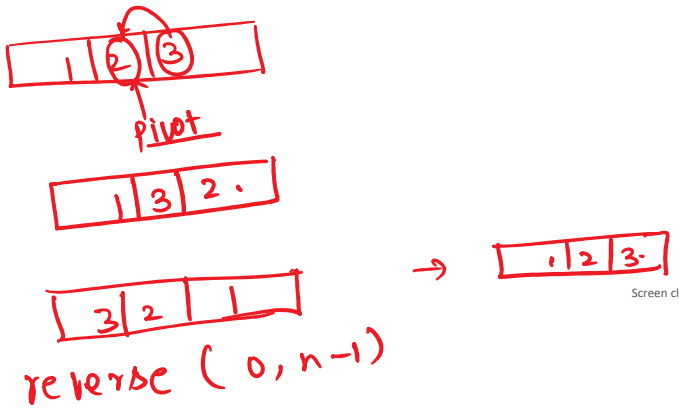
swap (pivot, successor)

reverse (pivot+1, n)

```
class Solution {
  static void reverse(int[] arr, int start, int end)
  {
    while (start < end) {
      swap(arr, start, end);
      start++;
      end--;
    }
```

```
{
  while (start < end) {
    swap(arr, start, end);
    start++;
    end--;
  }
}

static void swap(int[] arr, int i, int j)
{
  int temp = arr[i];
  arr[i] = arr[j];
  arr[j] = temp;
}

  public void nextPermutation(int[] nums) {
      int i, j;
      int n = nums.length;
      for( i=n-2;i>=0;i--){        //find pivot
          if(nums[i] < nums[i+1]){
              break;
          }

      }
      if(i<0){   //pivot not found reverse whole array
          reverse(nums, 0 , n - 1);
      }
      else{     //pivot found find successor
          for( j= n-1;j>=0;j--){
              if(nums[j] > nums[i]){
                  break;
              }
          }
          swap(nums, i, j);    //swap pivot and successor
          reverse(nums, i + 1, nums.length - 1);
      }
```

Screen clipping taken: 22-12-2022 23:38



Screen clipping taken: 22-12-2022 23:46

Leaders in an array

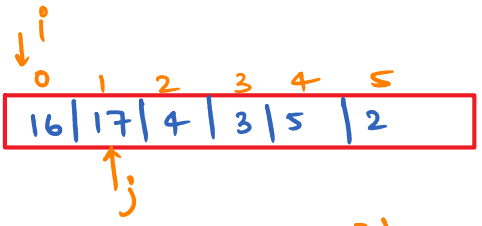
Easy Accuracy: 29.94% Submissions: 402K+ Points: 2

Bag Offers from Top Product Companies. Explore Exclusive Problems Now!

Given an array A of positive integers. Your task is to find the leaders in the array. An element of array is leader if it is greater than or equal to all the elements to its right side. The rightmost element is always a leader.

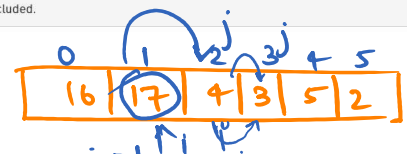
Example 1:

Input:
n = 6
A[] = {16,17,4,3,5,2}
Output: 17 5 2
Explanation: The first leader is 17 as it is greater than all the elements to its right. Similarly, the next leader is 5. The right most element is always a leader so it is also included.



bool leader = true.

if(arr[j] > arr[i])
leader = false;
break;
if (leader)
arr[i]



arr[0] > arr[i] → True. 17>16 → T
leader → false.
leader=false.

i=0 , j=1
17>16

ans → 17

4>17 → false.
3>4 → false
True.
5>3 → false
break
2>5 → false
↓
Add this

Ans → 17 True.

Screen clipping taken: 23-12-2022 00:00

TC:- O(n²)
SC:- O(1)

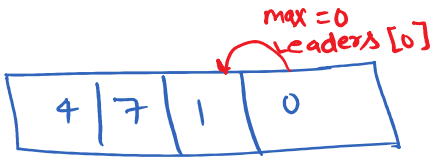
```
class Solution{
    //Function to find the leaders in the array.
    static ArrayList<Integer> leaders(int arr[], int n){
        // Your code here
        ArrayList<Integer>ans  = new ArrayList<Integer>();
            for (int i = 0; i < n - 1; i++) {
              boolean leader = true;

              //Checking whether arr[i] is greater than all the elements in its right si
              for (int j = i + 1; j < n; j++)
                if (arr[j] > arr[i]) {
                  leader = false;
                  break;
                }

              if (leader)
                ans.add(arr[i]);
            }
            ans.add(arr[n - 1]);
          return ans;
    }
}
```

Optimise Approach :-

```
class Solution{
    //Function to find the leaders in the array.
    static ArrayList<Integer> leaders(int arr[], int n){
        // Your code here

        ArrayList<Integer> array = new ArrayList<Integer>();

        int current_leader=arr[n-1];

        array.add(arr[n-1]);

        for(int i=n-2;i>=0;i--){

            if(current_leader<=arr[i]){

                current_leader=arr[i];

                array.add(current_leader);

            }

        }

        Collections.reverse(array);

        return (array);
    }
}
```

Screen clipping taken: 23-12-2022 00:09

TC :- O(n)
SC :- O(1)



1 > 0 → max = 1

Leader = [0, 1]

7 > 1 → max = 7

Leader = [0, 1, 7]

4 < 7 → max = 7

Leader = [0, 1, 7]

que :-

Longest consecutive subsequence

Medium Accuracy: **33.0%** Submissions: **209K+** Points: **4**

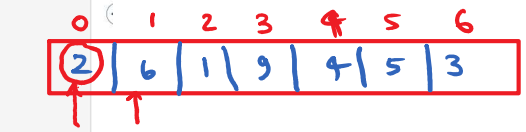
Bag Offers from Top Product Companies. Explore Exclusive Problems Now!

Given an array of positive integers. Find the length of the longest sub-sequence such that elements in the subsequence are consecutive integers, the **consecutive numbers can be in any order.**

Example 1:

Input:
N = 7
a[] = {2,6,1,9,4,5,3}
Output:
6
Explanation:
The consecutive numbers here are 1, 2, 3, 4, 5, 6. These 6 numbers form the longest consecutive subsequence.

Screen clipping taken: 23-12-2022 09:03

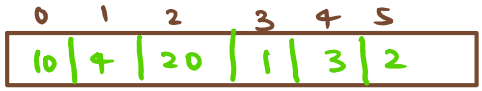


sort →

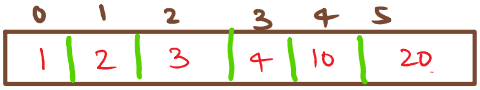
TC :- O(n log n) + O(n) + O(n)

Bruteforce Approach :-

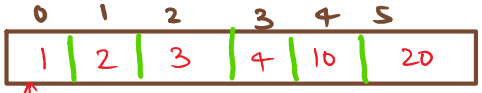
nums



sorting.



1st itr →

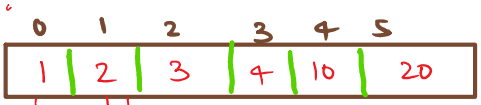


longest = 1 , curr = 1

arr[i] == arr[0] + 1
2 = 2 → cnt++

longest = 1, curr = 1

2nd itr :-

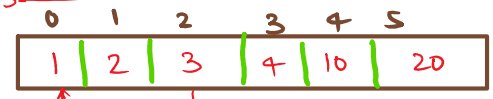


longest = 2, curr = 2.

Start i = 1

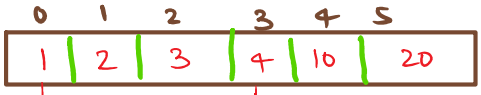
```
if( num[i] != num[i-1] )
    if ( num[i] == num[i-1] + 1 )
        curr++;
    else.
        longest = Math. Max ( longest, curr)
    curr = 1
return max(longest, curr)
```

3rd itr :-



longest = 3, curr = 3

4th itr :-



longest = 4, curr = 4.

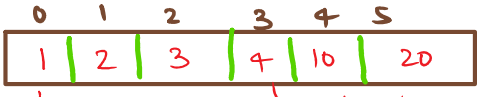
5th itr :-

longest = 4, curr = 1

→ else condition reach.

curr = 1

6th itr



longest = 4, curr = 2.

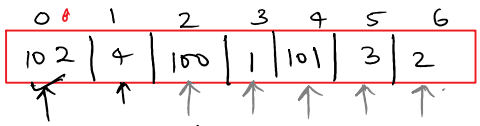
longest > curr

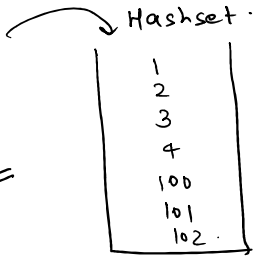
return = 4

```
class Solution
{
    // arr[] : the input array
    // N : size of the array arr[]

    //Function to return length of longest subsequence of consecutive integers.
    static int findLongestConseqSubseq(int arr[], int N)
    {
        // add your code here
        Arrays.sort(arr);
        int longest = 1;
        int curr = 1;
        for(int i=1;i<N;i++){
            if(arr[i] != arr[i-1]){
                if(arr[i] == arr[i-1] +1){
                    curr++;
                }
                else{
                    longest  = Math.max(longest, curr);
                    curr =1;
                }
            }
        }
        return Math.max(longest, curr);
    }
}
```

taken: 23-12-2022 09:26

Efficient Approach :-





102 → check 102-1 exist = 101 ✓

4 → check 4-1 exist = 3 ✓

100 → check 100-1 99 exist ✗

check for next no.

101 → ✓

102 ✓ exist

103 ✗ not exist

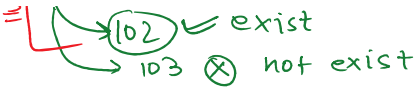
max = 3

now move to 1 → 1-1 = 0 ✗

length = 4

next 2 → 3 → 4 → 5 ✗

greater than previously Update longest = 4



greater than previously
Update longest = ?

TC :- O(N) + O(N) + O(N) = O(3N) → O(N)

SC :- O(N)

```java
class Solution {
    public int longestConsecutive(int[] nums) {
        HashSet<Integer> hs = new HashSet<Integer>();
        for(int num : nums){      //sbko hasset me enter kiya
            hs.add(num);
        }
        int longeststreak = 0;
        for(int num : nums){

            if(!hs.contains(num -1)){    //check usse ek km exist krta hai kya
                int currnum = num;

                int currstr = 1;
                while(hs.contains(currnum +1)){    //agr ek jyada exist krta hai to
                    currnum +=1;

                    currstr +=1;

                }
                longeststreak = Math.max(longeststreak, currstr);

            }
        }
        return longeststreak;
    }
}
```

Screen clipping taken: 23-12-2022 09:52

que :- set Matrix Zero :-

73. Set Matrix Zeroes Hint

Medium 10.1K 585

Companies

Given an `m x n` integer matrix `matrix`, if an element is `0`, set its entire row and column to `0`'s.

You must do it in place.

Example 1:

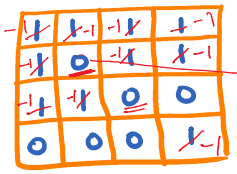
1	1	1		1	0	1
1	0	1		0	0	0
1	1	1		1	0	1

Input: matrix = [[1,1,1],[1,0,1],[1,1,1]]
Output: [[1,0,1],[0,0,0],[1,0,1]]

Screen clipping taken: 23-12-2022 10:12

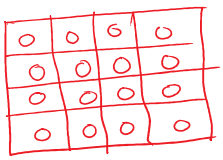
① If (
mat[i][j] == 0
ind = i-1
while(ind >= 0)
if (mat[ind][j] != 0)
{ mat[ind][j] = -1;
}
ind--;
}

bruteforce Approach :-



dont change 0
mat[i][j] == 0

Whenever there is -1 change to 0



TC :- (N*m) + (N+M)

SC :- O(1)

ind = i+1
while(ind < rows)
{ -1
}
ind++

ind j-1
↳
ind-- -1

if(matrix[i][i] <= 0
matrix[i][j] = 0
}

```java
class Solution {
    public void setZeroes(int[][] matrix) {
        int rows = matrix.length, cols = matrix[0].length;
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                if (matrix[i][j] == 0) {

                    int ind = i - 1;
                    while (ind >= 0) {
                        if (matrix[ind][j] != 0) {
                            matrix[ind][j] = -1;
                        }
```

```
                    ind--;
                }
                ind = i + 1;
                while (ind < rows) {
                    if (matrix[ind][j] != 0) {
                        matrix[ind][j] = -1;
                    }
                    ind++;
                }
                ind = j - 1;
                while (ind >= 0) {
                    if (matrix[i][ind] != 0) {
                        matrix[i][ind] = -1;


                    }
                    ind--;
                }
                ind = j + 1;
                while (ind < cols) {
                    if (matrix[i][ind] != 0) {
                        matrix[i][ind] = -1;


                    }
                    ind++;
                }
            }
        }
    }
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            if (matrix[i][j] <= 0) {
                matrix[i][j] = 0;
            }
        }
    }
}
}
```


Optimize Approach :- Using a single dummy :-

```
static void setZeroes(int[][] matrix) {
    int col0 = 1, rows = matrix.length, cols = matrix[0].length;

    for (int i = 0; i < rows; i++) {
        if (matrix[i][0] == 0) col0 = 0;
        for (int j = 1; j < cols; j++)
            if (matrix[i][j] == 0)
                matrix[i][0] = matrix[0][j] = 0;
    }

    for (int i = rows - 1; i >= 0; i--) {
        for (int j = cols - 1; j >= 1; j--)
            if (matrix[i][0] == 0 || matrix[0][j] == 0)
                matrix[i][j] = 0;
        if (col0 == 0) matrix[i][0] = 0;
    }
}
```

row = 0
col = 0
last row
last col
row = 0 || col = = 0

Qve :- Rotate 90° Matrix :-

Rotate by 90 degree

Easy Accuracy: **56.88%** Submissions: **53K+** Points: **2**

Given a square matrix of size **N x N**. The task is to rotate it by **90 degrees in anti-clockwise** direction without using any extra space.

Example 1:

Input:
N = 3
matrix[][] = {{1, 2, 3},
{4, 5, 6}
{7, 8, 9}}
Output:
Rotated Matrix:
3 6 9
2 5 8
1 4 7

Approach :-

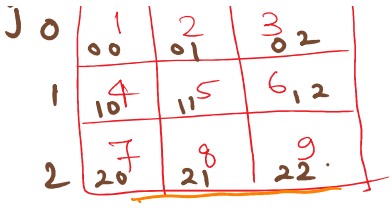
	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9

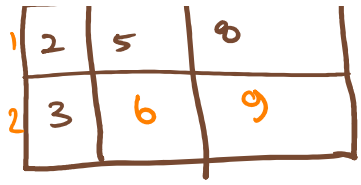
arr[i][j] = arr[j][i]

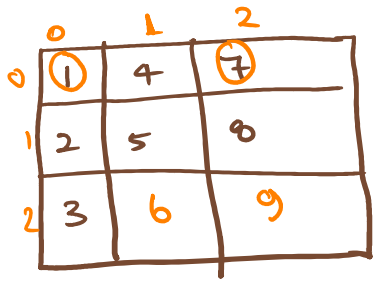
i=0	1	2
j 0	1 (00)	2 (01)
1	4	5

Transpose ⇒

	0	1	2
0	1	4	7
1	2	5	8







start = 0
end = n-1