Important Array Question

10 April 2025 17:10

1-> Longest Subarray with sum K | [Postives and Negatives]
2-> Kadane's Algorithm : Maximum Subarray Sum in an Array
3-> Count Subarray sum Equals K

Largest Element in an Array

06 October 2024 00:10

```
public static int largest(int[] arr)
  {
    int ans=arr[0];
    for(int i=1;i<arr.length;i++)
    {
      ans=Math.max(ans,arr[i]);
    }
    return ans;
    // code here
  }
```

Second Largest Element in an Array without sorting

06 October 2024 00:11

```
class Solution {
   public int getSecondLargest(int[] arr)
   {
      if(arr.length==1)
      {
         return -1;
      }
      int ans=arr[0];
      int res=-1;
      for(int i=1;i<arr.length;i++)
      {
         if(arr[i]>ans)
         {

           res=ans;

            ans=arr[i];
         }
         else
         {
            if(arr[i]>res&&arr[i]!=ans)
            {
               res=arr[i];
            }
         }


      }
      return res;
      // Code Here
   }
}
```

Check if array is sorted

06 October 2024 00:11

```
public class Solution {

  public boolean check(int[] nums) {
    int breakPointCount = 0;

    // Traverse the array and count how many times the sequence decreases
    for (int i = 0; i < nums.length - 1; i++) {
      if (nums[i] > nums[i + 1]) {
        breakPointCount++;
      }
    }

    // If there is more than 1 break, it's impossible to rotate and sort the array
    if (breakPointCount > 1) {
      return false;
    }

    // If there are no breaks, or exactly one, check if the array can be sorted by rotation
    // If no break, it's already sorted, so we return true
    // If one break, ensure the first element is greater than or equal to the last element
    return breakPointCount == 0 || nums[nums.length - 1] <= nums[0];
  }

  public static void main(String[] args) {
    Solution solution = new Solution();

    // Test case: Array can be sorted by rotation
    int[] nums1 = {3, 4, 5, 1, 2};
    System.out.println(solution.check(nums1));  // Output: true

    // Test case: Array cannot be sorted by rotation
    int[] nums2 = {2, 1, 3, 4};
    System.out.println(solution.check(nums2));  // Output: false

    // Test case: Array cannot be sorted by rotation (input provided by you)
    int[] nums3 = {2, 7, 4, 1, 2, 6, 2};
    System.out.println(solution.check(nums3));  // Output: false
  }
}
```

Remove duplicates from Sorted array

06 October 2024 00:11

```
public int removeDuplicates(int[] nums)
    {
        Set<Integer>s=new LinkedHashSet<>();
        for(int i=0;i<nums.length;i++)
        {
            s.add(nums[i]);
        }
        int n= s.size();
        int i=0;
        for(int a:s)
        {
            nums[i]=a;
            i++;
        }
        return n;
    }
```

Left Rotate an array by one place

06 October 2024 00:11

```
static void solve(int arr[], int n) {
  int temp = arr[0]; // storing the first element of array in a variable
  for (int i = 0; i < n - 1; i++) {
    arr[i] = arr[i + 1];
  }
  arr[n - 1] = temp; // assigned the value of variable at the last index
  return;

}
```

Left rotate an array by D places

06 October 2024 00:12

# Extra SPACE LAGEGA KAAKE

First half reverse+second half reverse+whole array reverse

```
public void rotate(int[] arr, int k1)
    {

        int n=arr.length;
        k1=k1%n;
        k1=n-k1;
        int[] temp=new int[k1];
        int i;
        for(i=0;i<k1;i++)
        {
            temp[i]=arr[i];
        }
        int j=0;
        for(int k=i;k<arr.length;k++)
        {
            arr[j]=arr[k];
            j++;
        }
        i=0;
        for(int k=j;k<arr.length;k++)
        {
            arr[k]=temp[i];
            i++;
        }
        return;
    }
```

Move Zeros to end

06 October 2024 00:12

```
private void swap(int[] nums, int i, int j)
    {
        int temp = nums[i];
        nums[i] = nums[j];
        nums[j] = temp;
    }
    public void moveZeroes(int[] nums)
    {
        int i=0;
        for(int j=0;j<nums.length;j++)
        {
            if(nums[j]!=0)
            {
                swap(nums,i,j);
                i++;
            }

        }
        for(int k=i;k<nums.length;k++)
        {
            nums[k]=0;
        }
        return;
    }
```

Linear Search

06 October 2024 00:12

```
static boolean searchInSorted(int arr[], int num)
   {

      int i;
      for(i=0;i<arr.length;i++)
      {
         if(arr[i]==num)
         return true;
      }
      return false;
   }
```

Find the Union

06 October 2024 00:12

```
import java.util.ArrayList;

public class Solution {

  public static ArrayList<Integer> findUnion(int a[], int b[]) {
    ArrayList<Integer> ans = new ArrayList<>();

    int i = 0;
    int j = 0;
    int m = a.length;
    int n = b.length;

    while (i < m && j < n) {
      // If both arrays have the same element, add it to the result
      if (a[i] == b[j]) {
        ans.add(a[i]);
        // Skip duplicates in array a
        while (i + 1 < m && a[i] == a[i + 1]) {
          i++;
        }
        // Skip duplicates in array b
        while (j + 1 < n && b[j] == b[j + 1]) {
          j++;
        }
        // Move both pointers forward
        i++;
        j++;
      }
      // If element in a is smaller, add it to the result
      else if (a[i] < b[j]) {
        ans.add(a[i]);
        // Skip duplicates in array a
        while (i + 1 < m && a[i] == a[i + 1]) {
          i++;
        }
        i++;
      }
      // If element in b is smaller, add it to the result
      else {
        ans.add(b[j]);
        // Skip duplicates in array b
        while (j + 1 < n && b[j] == b[j + 1]) {
          j++;
        }
        j++;
      }
    }

    // Add remaining elements from array a
    while (i < m) {
      ans.add(a[i]);
```

```
            while (i + 1 < m && a[i] == a[i + 1]) {
                i++;
            }
            i++;
        }

        // Add remaining elements from array b
        while (j < n) {
            ans.add(b[j]);
            while (j + 1 < n && b[j] == b[j + 1]) {
                j++;
            }
            j++;
        }

        return ans;
    }

    public static void main(String[] args) {
        int[] a = {1, 2, 4, 5, 6};
        int[] b = {2, 3, 5, 7};

        ArrayList<Integer> result = findUnion(a, b);
        System.out.println(result); // Output: [1, 2, 3, 4, 5, 6, 7]
    }
}
```

Find missing number in an array

06 October 2024 00:12

```
public int missingNumber(int[] nums)
    {
        int sum=0;
        for(int i=0;i<nums.length;i++)
        {
            sum+=nums[i];
        }
        int n=nums.length;
        n=n*(n+1)/2; //isse n me sum 0,1,2,3 in sab ka hoga
        return n-sum;

    }
```

Maximum Consecutive Ones

06 October 2024 00:13

```
public int findMaxConsecutiveOnes(int[] nums)
    {
        int cnt1=0;
        int ans=0;
        for(int i=0;i<nums.length;i++)
        {
            if(nums[i]==1)
            {
                cnt1++;
            }
            else
            {
                cnt1=0;
            }
            ans=Math.max(ans,cnt1);
        }
        return ans;
    }
```

Find the number that appears once, and other numbers twice.

06 October 2024 00:13

```
public int singleNumber(int[] nums)
    {
        int ans=nums[0];
        for(int i=1;i<nums.length;i++)
        {
            ans=ans^nums[i];
        }
        return ans;

    }
```

Longest subarray with given sum K(positives)***

06 October 2024 00:13

Take care of test case 2,0,0,3
k=3

If only 0 and positive integer is giver sliding window approch will work

====> ORDER MAP USE O(N*LOGN)
====> UNORDER MAP USE O(N);

```
import java.util.HashMap;
import java.util.Map;

public class Solution {

    public int longestSubarray(int[] arr, int k) {
        int sum = 0;
        int ans = 0;
        Map<Integer, Integer> mp = new HashMap<>();

        // Initialize map with the base case: sum 0 occurs at index -1
        mp.put(0, -1); // This handles the case where the sum equals k from the beginning

        for (int i = 0; i < arr.length; i++) {
            sum += arr[i];

            // If sum == k, we found a subarray from index 0 to i
            if (sum == k) {
                ans = Math.max(ans, i + 1); // Length from index 0 to i
            }

            // If sum - k has been seen before, it means a subarray exists with sum k
            if (mp.containsKey(sum - k)) {
                int value = mp.get(sum - k);
                ans = Math.max(ans, i - value);
            }

            // Store the first occurrence of this sum
            if (!mp.containsKey(sum)) {
                mp.put(sum, i);
            }
        }

        return ans;
    }

    public static void main(String[] args) {
        Solution solution = new Solution();
        int[] arr = {1, 2, 3, 4, 5};
        int k = 9;
        int result = solution.longestSubarray(arr, k);
        System.out.println(result); // Output: 2 (subarray [4, 5])
    }
}
```

Longest subarray with sum K (Positives + Negatives)

06 October 2024 00:13

SAME PREVIOUS CODE WILL WORK HERE ALSO

```
import java.util.HashMap;
import java.util.Map;

public class Solution {

    public int longestSubarray(int[] arr, int k) {
        int sum = 0;
        int ans = 0;
        Map<Integer, Integer> mp = new HashMap<>();

        // Initialize map with the base case: sum 0 occurs at index -1
        mp.put(0, -1); // This handles the case where the sum equals k from the beginning

        for (int i = 0; i < arr.length; i++) {
            sum += arr[i];

            // If sum == k, we found a subarray from index 0 to i
            if (sum == k) {
                ans = Math.max(ans, i + 1); // Length from index 0 to i
            }

            // If sum - k has been seen before, it means a subarray exists with sum k
            if (mp.containsKey(sum - k)) {
                int value = mp.get(sum - k);
                ans = Math.max(ans, i - value);
            }

            // Store the first occurrence of this sum
            if (!mp.containsKey(sum)) {
                mp.put(sum, i);
            }
        }

        return ans;
    }

    public static void main(String[] args) {
        Solution solution = new Solution();
        int[] arr = {1, 2, 3, 4, 5};
        int k = 9;
        int result = solution.longestSubarray(arr, k);
        System.out.println(result); // Output: 2 (subarray [4, 5])
    }
}
```

2Sum Problem

15 February 2025 19:53

```
class Solution
{
    public int[] twoSum(int[] nums, int target)
    {
          vector<int>v;

              map<int,int>mp;
              for(int i=0;i<nums.size();i++)
              {
                 if(mp.find(target-nums[i])!=mp.end())
                 {

                    v.push_back(mp.find(target-nums[i])->second);
                    v.push_back(i);
                    return v;
                 }
                 else
                 {
                    mp[nums[i]]=i;
                 }
              }
              return v;


    }
}
```

JAVA

```
public int[] twoSum(int[] nums, int target)
    {
        int[]arr=new int[2];
        Map<Integer,Integer>mp=new HashMap<>();
        for(int i=0;i<nums.length;i++)
        {
            int complement=target-nums[i];
            if (mp.containsKey(complement))
            {
                arr[0]=mp.get(complement);
                arr[1]=i;
                return arr;
            }
            else
            {
                mp.put(nums[i],i);
            }
        }
        return arr;
    }
```

Sort an array of 0's 1's and 2's

15 February 2025 20:05

```
public class Solution {

   // Swap method that works with array indices
   public static void swap(int[] nums, int i, int j) {
      int temp = nums[i];
      nums[i] = nums[j];
      nums[j] = temp;
   }

   // Sort the array for Dutch National Flag problem (0s, 1s, and 2s)
   public void sortColors(int[] nums) {
      int i = 0;  // Pointer for the 0's region
      int j = 0;  // Pointer for the current element being checked
      int k = nums.length - 1;  // Pointer for the 2's region

      while (j <= k) {
         if (nums[j] == 1) {
            j++;  // 1's are in the middle, just move forward
         } else if (nums[j] == 0) {
            swap(nums, i, j);  // Swap 0 to the left side
            i++;
            j++;
         } else {
            swap(nums, j, k);  // Swap 2 to the right side
            k--;
         }
      }
   }

   public static void main(String[] args) {
      Solution solution = new Solution();
      int[] nums = {2, 0, 1, 2, 1, 0};

      solution.sortColors(nums);

      // Output the sorted array
      for (int num : nums) {
         System.out.print(num + " ");
      }
   }
}
```

Majority Element (>n/2 times)

15 February 2025 20:16

```
public int majorityElement(int[] nums)
    {
        int a=nums[0];
        int cnt=1;
        for(int i=1;i<nums.length;i++)
        {
            if(nums[i]==a)
            {
                cnt++;
            }
            else
            {
                cnt--;
            }
            if(cnt==0)
            {
                a=nums[i];
                cnt=1;
            }
        }
        int count=0;
        for(int i=0;i<nums.length;i++)
        {
            if(nums[i]==a)
            {
                count++;
            }
        }
        if(count>=nums.length/2)
        {
            return a;
        }
        return -1;

    }
```

Kadane's Algorithm, maximum subarray sum

15 February 2025 21:07

```
int maxSubArray(vector<int>& nums)
    {
        int maxsum=nums[0];
        int sum=0;

        for(int i=0;i<nums.size();i++)
        {
            if(nums[i]>0)
            {
                sum=sum+nums[i];
                maxsum=max(maxsum,sum);
            }
            else
            {
                sum=sum+nums[i];
                maxsum=max(maxsum,sum);
                if(sum<0)
                {
                    sum=0;
                }
            }
        }
        return maxsum;

    }
```

Print subarray with maximum subarray sum (extended version of above problem)*****0

15 February 2025 21:39

Stock Buy and Sell

15 February 2025   21:49

```
public int maxProfit(int[] arr)
    {
        int mn=arr[0];
        int n=arr.length;
        int ans=0;
        for(int i=1;i<n;i++)
        {
            ans=Math.max(arr[i]-mn,ans);
            if(arr[i]<mn)
            {
                mn=arr[i];
            }
        }
        return ans;

    }
```

Rearrange the array in alternating positive and negative items

15 February 2025 21:50

Ye O(1) space me nahi solve hoga

```
import java.util.ArrayList;
import java.util.List;

public class Solution {

    public List<Integer> rearrangeArray(int[] arr) {
        List<Integer> pos = new ArrayList<>();
        List<Integer> neg = new ArrayList<>();

        // Separate positive and negative numbers
        for (int i = 0; i < arr.length; i++) {
            if (arr[i] >= 0) {
                pos.add(arr[i]);
            } else {
                neg.add(arr[i]);
            }
        }

        List<Integer> result = new ArrayList<>();
        int i = 0, j = 0;

        // Interleave positive and negative numbers
        while (i < pos.size() && j < neg.size()) {
            result.add(pos.get(i));
            i++;
            result.add(neg.get(j));
            j++;
        }

        // Add remaining positive numbers (if any)
        while (i < pos.size()) {
            result.add(pos.get(i));
            i++;
        }

        // Add remaining negative numbers (if any)
        while (j < neg.size()) {
            result.add(neg.get(j));
            j++;
        }

        return result;
    }

    public static void main(String[] args) {
        Solution solution = new Solution();

        int[] arr = {1, -2, 3, -4, 5, -6};
        List<Integer> result = solution.rearrangeArray(arr);

        // Print the result
        for (int num : result) {
            System.out.print(num + " ");
        }
    }
}
```

Next Permutation

15 February 2025 22:47

Input: arr = [2, 4, 1, 7, 5, 0]
Output: [2, 4, 5, 0, 1, 7]

```
import java.util.*;

public class Solution {

   public void nextPermutation(int[] nums) {
      int i = nums.length - 1;

      // Find the rightmost element that is smaller than its next element
      while (i > 0 && nums[i] <= nums[i - 1]) {
         i--;
      }

      // If the entire array is in descending order, reverse it to get the smallest permutation
      if (i == 0) {
         reverse(nums, 0, nums.length - 1);
      } else {
         i--;  // Move i to the correct position of the smaller number

         int j = nums.length - 1;

         // Find the number that is larger than nums[i] and swap them
         while (nums[j] <= nums[i]) {
            j--;
         }

         // Swap the elements at indices i and j
         swap(nums, i, j);

         // Reverse the subarray after i to get the next permutation
         reverse(nums, i + 1, nums.length - 1);
      }
   }

   // Swap function to swap elements in the array
   private void swap(int[] nums, int i, int j) {
      int temp = nums[i];
      nums[i] = nums[j];
      nums[j] = temp;
   }

   // Reverse a portion of the array from index `start` to `end`
   private void reverse(int[] nums, int start, int end) {
      while (start < end) {
         swap(nums, start, end);
         start++;
```

```
            end--;
        }
    }

    public static void main(String[] args) {
        Solution solution = new Solution();
        int[] nums = {1, 2, 3};

        solution.nextPermutation(nums);

        // Print the result after the next permutation
        System.out.println(Arrays.toString(nums));
    }
}
```

Array Leaders

15 February 2025 22:58

```
import java.util.*;

public class Solution {

    public List<Integer> leaders(int[] arr) {
        List<Integer> ans = new ArrayList<>();
        int n = arr.length;

        // Start by adding the last element to the leaders list
        ans.add(arr[n - 1]);
        int t = arr[n - 1]; // This is the current leader

        // Traverse the array from the second last element to the first
        for (int i = n - 2; i >= 0; i--) {
            if (arr[i] >= t) {
                ans.add(arr[i]);
                t = arr[i]; // Update the current leader
            }
        }

        // Reverse the list to maintain the correct order of leaders
        Collections.reverse(ans);

        return ans;
    }

    public static void main(String[] args) {
        Solution solution = new Solution();
        int[] arr = {16, 17, 4, 3, 5, 2};

        List<Integer> result = solution.leaders(arr);

        // Print the leaders
        System.out.println(result);
    }
}
```

Longest Consecutive Sequence in an Array

15 February 2025 23:00

```
public int longestConsecutive(int[] arr)
  {

       Set<Integer>s=new HashSet<>();
       for(int i=0;i<arr.length;i++)
       {
         s.add(arr[i]);
       }
       int ans=1;

       for(int i=0;i<arr.length;i++)
       {
         int a=arr[i]-1;
         int cnt=1;
         while(s.contains(a))
         {
           cnt++;
           a--;
         }
         ans=Math.max(ans,cnt);
       }
       return ans;
  }
```

```
public static int longestSuccessiveElements(int[] a) {
        int n = a.length;
        if (n == 0)
            return 0;
int longest = 1;
        Set<Integer> set = new HashSet<>();
// put all the array elements into set
        for (int i = 0; i < n; i++) {
            set.add(a[i]);
        }
// Find the longest sequence
        for (int it : set) {
            // if 'it' is a starting number
            if (!set.contains(it - 1)) {
                // find consecutive numbers
                int cnt = 1;
                int x = it;
                while (set.contains(x + 1)) {
                    x = x + 1;
                    cnt = cnt + 1;
                }
                longest = Math.max(longest, cnt);
            }
        }
        return longest;
    }
```

From <https://takeuforward.org/data-structure/longest-consecutive-sequence-in-an-array/>

Set Matrix Zeros

16 February 2025 09:56

```
public void setZeroes(int[][] matrix)
    {
        int m=matrix.length;
        int n=matrix[0].length;
        int[] row=new int[m];
        int[] col=new int[n];
        for(int i=0;i<m;i++)
        {
            for(int j=0;j<n;j++)
            {
               if(matrix[i][j]==0)
               {
                    row[i]=1;
                    col[j]=1;
               }
            }
        }
        //Modify
        for(int i=0;i<m;i++)
        {
            for(int j=0;j<n;j++)
            {
               if(row[i]==1||col[j]==1)
               {
                    matrix[i][j]=0;
               }
            }
        }
        return;

    }
```

Rotate Matrix by 90 degrees

15 February 2025 23:55

```
void rotateby90(vector<vector<int> >& a, int n)
  {
     for (int row = 0; row < n; row++)
    {
               for (int col = row+1; col < n; col++)
                {
                       /**swap a[col][row] and a[row][col]**/
                       a[row][col]^=a[col][row];
                       a[col][row]^=a[row][col];
                       a[row][col]^=a[col][row];
               }
       }

       /** Reverse columns **/
       for(int i=0;i<n;i++)
       {
          for(int j=0;j<(n/2);j++)
       {
             /**swap(a[j][i],a[n-1-j][i])**/
             a[j][i]^=a[n-1-j][i];
             a[n-1-j][i]^=a[j][i];
             a[j][i]^=a[n-1-j][i];
          }
       }
```

Spiral Matrix

16 February 2025 10:32

```
public List<Integer> spiralOrder(int[][] mat)
    {
        // Define ans list to store the result.
        List<Integer> ans = new ArrayList<>();

        int n = mat.length; // no. of rows
        int m = mat[0].length; // no. of columns

        // Initialize the pointers required for traversal.
        int top = 0, left = 0, bottom = n - 1, right = m - 1;
        // Loop until all elements are not traversed.
        while (top <= bottom && left <= right) {
            // For moving left to right
            for (int i = left; i <= right; i++)
                ans.add(mat[top][i]);
            top++;
            // For moving top to bottom.
            for (int i = top; i <= bottom; i++)
                ans.add(mat[i][right]);
            right--;
            // For moving right to left.
            if (top <= bottom) {
                for (int i = right; i >= left; i--)
                    ans.add(mat[bottom][i]);
                bottom--;
            }
            // For moving bottom to top.
            if (left <= right) {
                for (int i = bottom; i >= top; i--)
                    ans.add(mat[i][left]);
                left++;
            }
        }
        return ans;

    }
```

Count subarrays with given sum***

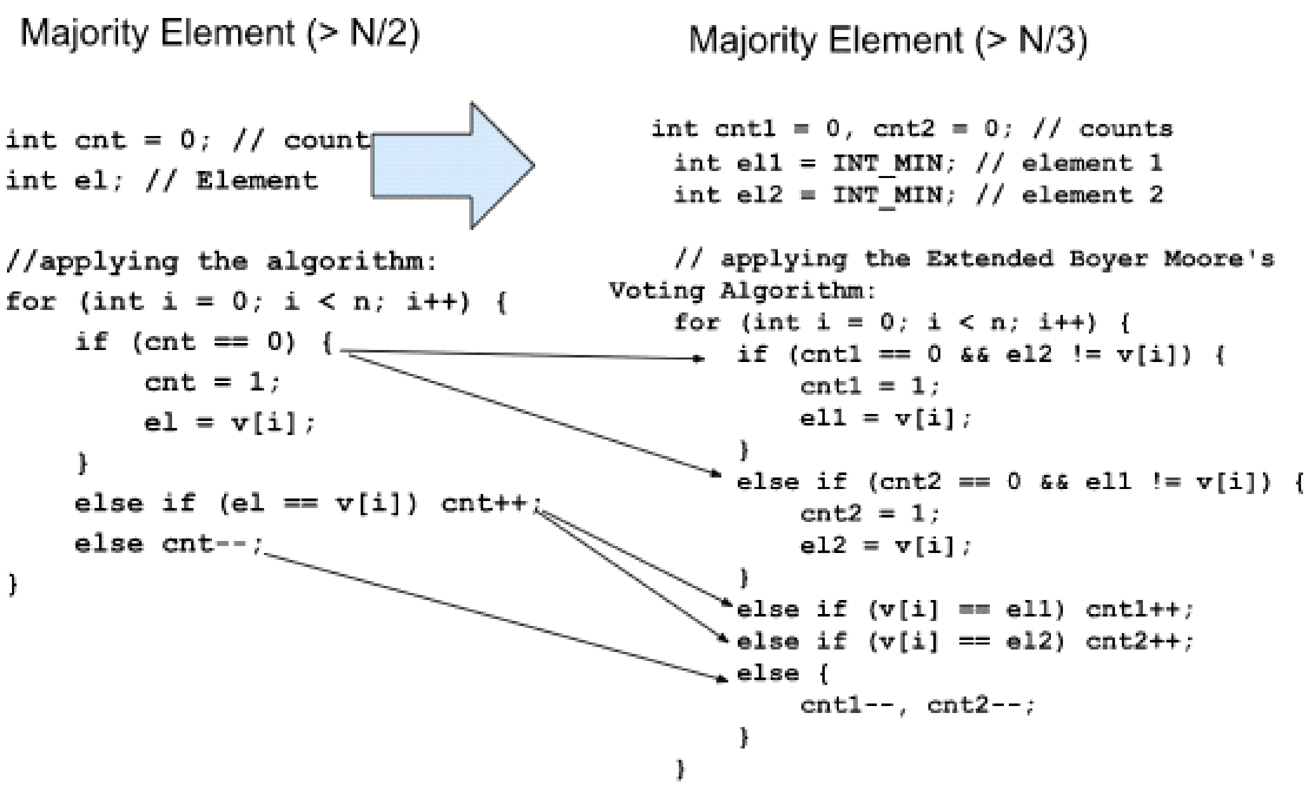
16 February 2025 10:33

Pascal's Triangle

17 February 2025 21:34

Majority Element (n/3 times)

17 February 2025 21:34



Majority Element (> N/2)

```
int cnt = 0; // count
int el; // Element

//applying the algorithm:
for (int i = 0; i < n; i++) {
    if (cnt == 0) {
        cnt = 1;
        el = v[i];
    }
    else if (el == v[i]) cnt++;
    else cnt--;
}
```

Majority Element (> N/3)

```
int cnt1 = 0, cnt2 = 0; // counts
 int el1 = INT_MIN; // element 1
 int el2 = INT_MIN; // element 2

 // applying the Extended Boyer Moore's
Voting Algorithm:
 for (int i = 0; i < n; i++) {
     if (cnt1 == 0 && el2 != v[i]) {
         cnt1 = 1;
         el1 = v[i];
     }
     else if (cnt2 == 0 && el1 != v[i]) {
         cnt2 = 1;
         el2 = v[i];
     }
     else if (v[i] == el1) cnt1++;
     else if (v[i] == el2) cnt2++;
     else {
         cnt1--, cnt2--;
     }
 }
```

3-Sum Problem

17 February 2025 21:34

```
 public static List<List<Integer>> triplet(int n, int[] arr) {
        List<List<Integer>> ans = new ArrayList<>();
        Arrays.sort(arr);
for (int i = 0; i < n; i++) {
            //remove duplicates:
            if (i != 0 && arr[i] == arr[i - 1]) continue;
//moving 2 pointers:
            int j = i + 1;
            int k = n - 1;
            while (j < k) {
                int sum = arr[i] + arr[j] + arr[k];
                if (sum < 0) {
                    j++;
                } else if (sum > 0) {
                    k--;
                } else {
                    List<Integer> temp = Arrays.asList(arr[i], arr[j], arr[k]);
                    ans.add(temp);
                    j++;
                    k--;
                    //skip the duplicates:
                    while (j < k && arr[j] == arr[j - 1]) j++;
                    while (j < k && arr[k] == arr[k + 1]) k--;
                }
            }
        }
return ans;
    }
```

4-Sum Problem

17 February 2025 21:34

```
int maxLen(int A[], int n)
  {
      // Your code here
      HashMap<Integer, Integer> mpp = new HashMap<Integer, Integer>();

      int maxi = 0;
      int sum = 0;

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

          sum += A[i];

          if(sum == 0) {
              maxi = i + 1;
          }
          else {
              if(mpp.get(sum) != null) {

                  maxi = Math.max(maxi, i - mpp.get(sum));
              }
              else {

                  mpp.put(sum, i);
              }
          }
      }
      return maxi;
  }
```

Largest Subarray with 0 Sum

17 February 2025 21:34

Count number of subarrays with given xor K

17 February 2025 21:34

Merge Overlapping Subintervals

17 February 2025 21:35

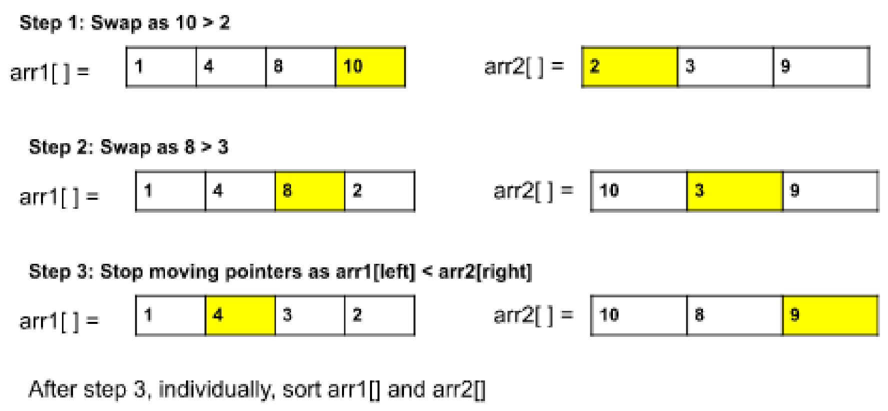
```
public static List<List<Integer>> mergeOverlappingIntervals(int[][] arr) {
        int n = arr.length; // size of the array
        //sort the given intervals:
        Arrays.sort(arr, new Comparator<int[]>() {
            public int compare(int[] a, int[] b) {
                return a[0] - b[0];
            }
        });

        List<List<Integer>> ans = new ArrayList<>();

        for (int i = 0; i < n; i++) {
            // if the current interval does not
            // lie in the last interval:
            if (ans.isEmpty() || arr[i][0] > ans.get(ans.size() - 1).get(1)) {
                ans.add(Arrays.asList(arr[i][0], arr[i][1]));
            }
            // if the current interval
            // lies in the last interval:
            else {
                ans.get(ans.size() - 1).set(1,
                                    Math.max(ans.get(ans.size() - 1).get(1), arr[i][1]));
            }
        }
        return ans;
    }
```

Merge two sorted arrays without extra space

17 February 2025 21:35



Step 1: Swap as 10 > 2

arr1[ ] = | 1 | 4 | 8 | 10 | arr2[ ] = | 2 | 3 | 9 |

Step 2: Swap as 8 > 3

arr1[ ] = | 1 | 4 | 8 | 2 | arr2[ ] = | 10 | 3 | 9 |

Step 3: Stop moving pointers as arr1[left] < arr2[right]

arr1[ ] = | 1 | 4 | 3 | 2 | arr2[ ] = | 10 | 8 | 9 |

After step 3, individually, sort arr1[] and arr2[]

Find the repeating and missing number

17 February 2025 21:35

Count Inversions

17 February 2025 21:35

Reverse Pairs

17 February 2025 21:35

Maximum Product Subarray

17 February 2025 21:36