

- Introduction to Arrays – C | C++ | Java
- Introduction to 2-D Arrays – C | C++ | Java
- Sorting of Array – C | C++ | Java
- Array Rotation – C | C++ | Java
- Reverse an array or string – C | C++ | Java
- Find pairs in array with given sum – C | C++ | Java
- Sort the array in Waveform – C | C++ | Java
- Majority Element in Array – C | C++ | Java
- Boyer-Moore's Voting Algorithm – C | C++ | Java
- K-pairs with smallest sum in 2 arrays – C | C++ | Java
- Largest Sum Contiguous SubArray – C | C++ | Java
- Maximum Average Sub-array of K length – C | C++ | Java
- Size of sub-array with max sum – C | C++ | Java
- Sub-array with given sum – C | C++ | Java
- Triplet that sum to a given value – C | C++ | Java
- Segregate 0's and 1's in array – C | C++ | Java
- Segregate 0's 1's and 2's in array – C | C++ | Java
- Sort elements in array by frequency – C | C++ | Java
- Finding pythagorean triplets in an array – C | C++ | Java
- Reorder array using given indexes – C | C++ | Java
- Merging two sorted arrays – C | C++ | Java
- Minimum number of Merge Operations to make an Array Palindrome – C | C++ | Java
- Find Zeros to be Flipped so that number of Consecutive 1's is maximized – C | C++ | Java

Ques1. Segregate 0's and 1's in array

Segregate 0s and 1s in an array | GeeksforGeeks

Method 1 (Count 0s or 1s)

- ❑ Count the number of 0s. Let count be C .
- ❑ Once we have count, we can put C 0s at the beginning and 1s at the remaining $(n - C)$ positions in array.
- ❑ Time Complexity: $O(n)$
- ❑ This method traverses the array two times.

Method 2 (Use two indexes to traverse)

- ❑ Maintain two indexes. Initialize first index *left* as 0 and second index *right* as $n-1$.
 - ❑ Do following while $left < right$
 - ❑ Keep incrementing index *left* while there are 0s at it
 - ❑ Keep decrementing index *right* while there are 1s at it
 - ❑ If $left < right$ then exchange $arr[left]$ and $arr[right]$
- Implementation:

Method 2 (Use two indexes to traverse)

```

/*Function to put all 0s on left and all 1s on right*/
void segregate0and1(int arr[], int size)
{
    /* Initialize left and right indexes */
    int left = 0, right = size-1;

    while (left < right)
    {
        /* Increment left index while we see 0 at left */
        while (arr[left] == 0 && left < right)
            left++;

        /* Decrement right index while we see 1 at right */
        while (arr[right] == 1 && left < right)
            right--;

        /* If left is smaller than right then there is a 1 at left
        and a 0 at right. Exchange arr[left] and arr[right]*/
        if (left < right)
        {
            arr[left] = 0;
            arr[right] = 1;
            left++;
            right--;
        }
    }
}

```

Ques2 .Segregate 0s,1s,2s without sort (complexity $O(n \log n)$ se kam ho)

Sort an array of 0s 1s 2s

Method - 2

```

while (m <= h)
{
    if (a[m] == 0)
    {
        swap(a[l], a[m])
        l++;
        m++;
    }
    if (a[m] == 1)
        m++;
    if (a[m] == 2)
    {
        swap(a[m], a[h])
        h--;
    }
}

```

$l \rightarrow 0$
 $m \rightarrow 0$
 $h \rightarrow (N-1)$

Ques3. Sort elements in array by frequency

* Sort the array based on frequency of values
 (Sort as per value if frequency is same)

arr = { 10, 7, 10, 11, 10, 7, 5, 6 }

↓

{ 10, 10, 10, 7, 7, 5, 6, 11 }

10, 10, 10, 7, 7, 11, 6, 5

10 → 3
 11 → 1
 7 → 2
 5 → 1
 6 → 1

→ mal

```
Array - 32: Sort array based on frequency of value (If frequency is same then sort based on value)
30 }
31
32 public static List<Integer> sortBasedOnFrequencyAndIndex(List<Integer> list) {
33     Map<Integer, Integer> map = new HashMap<>();
34
35     for (int i = 0; i < list.size(); i++) {
36         map.put(list.get(i), map.getOrDefault(list.get(i), 0) + 1);
37     }
38
39     Collections.sort(list, (k1, k2) -> {
40         int freq1 = map.get(k1);
41         int freq2 = map.get(k2);
42
43         if(freq1 != freq2) {
44             return freq2 - freq1;
45         }
46
47         return list.indexOf(k1) - list.indexOf(k2);
48     });
49
50     return list;
51 }
52
53 public static void main(String[] args) {
54     Integer[] arr = {10, 7, 10, 11, 10, 7, 6, 5};
55     List<Integer> list = Arrays.asList(arr);
56     System.out.println(ArrayApp.sortBasedOnFrequencyAndValue(list));
57 }
```

another solution is by creating max heap

Ques4.Sort the element of array in wave form

1.Method

Sort an array in wave form | GeeksforGeeks

Solution 1: Understanding the code

```
#include<iostream>
#include<algorithm>
using namespace std;

void swap(int *x, int *y)
{
    int temp = *x;
    *x = *y;
    *y = temp;
}

void sortInWave(int arr[], int n)
{
    sort(arr, arr+n);

    for (int i=0; i<n-1; i += 2)
        swap(&arr[i], &arr[i+1]);
}
```

```
int main()
{
    int arr[] = {10, 90, 49, 2, 1, 5, 23};
    int n = sizeof(arr)/sizeof(arr[0]);
    sortInWave(arr,n);
    for (int i=0; i<n; i++)
        cout << arr[i] << " ";
    return 0;
}
```

WORKING

After Sorting: {1, 2, 5, 10, 23, 49, 90}

After Swapping: {2, 1, 10, 5, 49, 23, 90}

Time Complexity: **$O(n \log n)$**

2.Method

Solution 2:

We can sort the array in wave form by doing a single traversal of given array.

IDEA

The idea is based on the fact that if we make sure that all even positioned (at index 0, 2, 4, ..) elements are greater than their adjacent odd elements, we don't need to worry about odd positioned element.

STEPS

- 1) Traverse all even positioned elements of input array, and do following.
 - a) If current element is smaller than previous odd element, swap previous and current.
 - b) If current element is smaller than next odd element, swap next and current.

Sort an array in wave form | GeeksforGeeks

Solution 2: Understanding the code

```
#include<iostream>
using namespace std;
void swap(int *x, int *y)
{
    int temp = *x;
    *x = *y;
    *y = temp;
}

void sortInWave(int arr[], int n)
{
    for (int i = 0; i < n; i+=2)
    {
        if (i>0 && arr[i-1] > arr[i] )
            swap(&arr[i], &arr[i-1]);

        if (i<n-1 && arr[i] < arr[i+1] )
            swap(&arr[i], &arr[i + 1]);
    }
}
```

```
int main()
{
    int arr[] = {10, 90, 49, 2, 1, 5, 23};
    int n = sizeof(arr)/sizeof(arr[0]);
    sortInWave(arr, n);
    for (int i=0; i<n; i++)
        cout << arr[i] << " ";
    return 0;
}
```

Time Complexity: **$O(n)$**

3:50 / 4:32


```

for i = 1 to n-1
{
    if arr[i] > arr[i-1]
        swap(arr,i,i-1)

    if arr[i] > arr[i+1] && i <= n-2
        swap(arr,i,i+1)

    i+=2
}

```

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

```

for i = 1 to n-1
{
    if arr[i] > arr[i-1]
        swap(arr,i,i-1)

    if arr[i] > arr[i+1] && i <= n-2
        swap(arr,i,i+1)

    i+=2
}

```

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

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

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

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

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

Ques5.Find Pairs in array with given sum

```

#include <stdio.h>
int main()
{
    int n;
    scanf("%d",&n);
    int array[n];
    int check;
    int sum, count = 0;
    for(int i=0; i<n; i++)
    {
        scanf("%d",&array[i]);
    }
    scanf("%d",&check);
}

```

```

for(int i=0; i<n; i++)
{
    for(int j=i+1; j<n; j++)
    {
        sum = array[i] + array[j];
        if(sum == check)
        {
            printf("[%d %d]\n",array[i],array[j]);
            count ++;
        }
    }
}
printf("Total Number of Pairs : %d",count);
return 0;
}

```

Another better approach

```

* For example -
*
* low = 0
* high = length - 1;
*
* while(low < high) {
*
*     if(arr[low] + arr[high] > sum) {
*         high--;
*     } else if (arr[low] + arr[high] < sum) {
*         low++;
*     } else if (arr[low] + arr[high] == sum) {
*         print
*         low++;
*         high--;
*     }
* }
* arr[] = {1, 2, 3, 4, 5, 6, 7};
* sum = 9
*
* Output:
*
* Pair (2 , 7 )
* Pair (3 , 6 )
* Pair (4 , 5 )

```

Ques6.Triplet that sum to a given value

* Find all triplets for given target Sum

arr = {1, 2, -3, 4, -2, -1}

Sum = 1

for (i = 0 to n-1) } $O(n^3)$
 for () }
 for (i = 0 to n-1)
 for (j = i+1 to n-1)
 for (k = j+1 to n-1)

[-1, 4, -2] [1, 2, -2]

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

1, -3 [4, -2, -1]

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

$O(n^3)$

Java code :

```
// Java program to find a triplet
class prepinsta {

    boolean findnumbers(int A[], int arrsize, int sum)
    {
        int l, r;

        for (int i = 0; i < arrsize - 2; i++) {

            for (int j = i + 1; j < arrsize - 1; j++) {

                for (int k = j + 1; k < arrsize; k++) {
                    if (A[i] + A[j] + A[k] == sum) {
                        System.out.print("Triplet is " + A[i] + ", " + A[j] + ", " + A[k]);
                        return true;
                    }
                }
            }
        }

        //no triplet
        return false;
    }

    public static void main(String[] args)
    {
        prepinsta triplet = new prepinsta();
        int A[] = {1, 2, 3, 4, 5};
        int sum = 9;
        int arrsize = A.length;

        triplet.findnumbers(A, arrsize, sum);
    }
}
```

Output:

Triplet is 5, 3, 1

Find a triplet that sum to a given value | GeeksforGeeks

```
// returns true if there is triplet with sum equal to the given sum
bool find3Numbers(int A[], int arr_size, int sum)
{
    int l, r;

    /* Sort the elements */
    sort(A, A+arr_size);

    /* Now fix the first element one by one and find the
    other two elements */
    for (int i=0; i<arr_size-2; i++)
    {
        // To find the other two elements, start two index
        // variables from two corners of the array and move
        // them toward each other
        l = i + 1; // index of the first element in the
        // remaining elements
        r = arr_size-1; // index of the last element
        while (l < r)
        {
            if ( A[l] + A[l] + A[r] == sum)
            {
                printf("Triplet is %d, %d, %d", A[l],
                    A[l], A[r]);
                return true;
            }
            else if (A[l] + A[l] + A[r] < sum)
                l++;
            else // A[l] + A[l] + A[r] > sum
                r--;
        }
    }

    // If we reach here, then no triplet was found
    return false;
}
```

Ques 7. Rotation of array

1. Left Rotation of array

#include <stdio.h>

```
int main() {
    int x[]={1,2,3,4,5};
    int size=5;
    int r=4;
    int num,i;
    while(r)
    {
        num=x[0];
        for(i=1;i<size;i++){ x[i-1]=x[i];}
        x[i-1]=num;
        r--;
    }
    for(i=0;i<size;i++) printf("%d\n",x[i]);
    return 0;
}
```

2. Right Rotation of array

#include <stdio.h>

```
int main() {

    int x[]={10,11,12,13,14,15,16};
    int size=7;
    int r=4;
```

```

int ei=size-1;
int num,i;
while(r)
{
    num=x[ei];
    for(i=ei;i>0;i--) x[i]=x[i-1];
    x[i]=num;
    r--;
}
for(i=0;i<size;i++) printf("%d\n",x[i]);
return 0;

```

Ques8.Majority Element in array

```

#include<stdio.h>
int main()
{
//int x[]={1,2,1,2,1,4,5,1,2};
//11 22 33 44 55 66 77 44
int x[]={11,22,33,44,55,66,77,44};
int size=8;
int count=0;
int element;
int largestElementCount=-1;
for(int i=0;i<size-1;i++)
{
    count=1;
    for(int j=i+1;j<size;j++)
    {
        if(x[i]==x[j])count++;
    }
    if(largestElementCount<count)
    {
        largestElementCount=count;
        element=x[i];
    }
}
printf("%d\n",element);

return 0;
}

```

Optimized solution of above problem

Boyer-Moore's Voting Algorithm

```

#include <stdio.h>
int majorityElement(int x[],int size)
{
    int majorityElement=-1;
    int count=0;

```

```

for(int i=0;i<size;i++)
{
    if(count==0)
    {
        majorityElement=x[i];
        count=1;
    }
    else if(majorityElement==x[i])
    {
        count++;
    }
    else
    {
        count--;
    }
}
return majorityElement;
}
int main() {
    int x[]={2,1,1,3,5,4,3,3,3};
    printf("%d\n",majorityElement(x,9));
    return 0;
}

```

Ques9.MaxSubArraySum

```

#include <stdio.h>
#include<limits.h>
int maxSumSubArray(int x[],int size)
{
    int maxSum=INT_MIN;
    int startIndex=0;
    int endIndex=0;
    int sum;
    for(int si=0;si<size;si++)
    {
        for(int ei=0;ei<size;ei++)
        {
            sum=0;
            for(int j=si;j<=ei;j++)
            {
                sum=sum+x[j];
            }
            if(sum>maxSum)
            {
                maxSum=sum;
                startIndex=si;
                endIndex=ei;
            }
        }
    }
}

```

```

    }
    printf("Start index :%d\n",startIndex);
    printf("End index :%d\n",endIndex);
    return maxSum;
}

int main() {
    int x[]={5,-4,-2,6,-1};
    printf("%d\n",maxSumSubArray(x,5));
    return 0;
}

```

Kaden's Algorithm

```

#include <stdio.h>
#include<limits.h>
int maxSumSubArray(int x[],int size)
{
    int maxSum=0;
    int currentSum=0;
    for(int i=0;i<size;i++)
    {
        currentSum=currentSum+x[i];
        if(currentSum<0) currentSum=0;
        if(currentSum>maxSum) maxSum=currentSum;
    }
    return maxSum;
}

int main() {
    int x[]={5,-4,-2,6,7};
    printf("%d\n",maxSumSubArray(x,5));
    return 0;
}

```

Ques9.Maximum Average subarray of k length.

```

#include <stdio.h>

#include<limits.h>

int maximumAverageSubArray(int x[],int size,int lengthOfSubArray)
{
    int i;

    int sum=0;

    for(i=0;i<lengthOfSubArray;i++) sum+=x[i];

    int maxSum=sum;
}

```

```

    for(i=lengthOfSubArray;i<size;i++)
    {
        sum=sum+x[i]-x[i-lengthOfSubArray];
        if(sum>maxSum)
        {
            maxSum=sum;
        }
    }

    return maxSum/lengthOfSubArray;
}

int main() {
    int x[]={11,-8,16,-7,24,-2,3};
    int lengthOfSubArray=3;
    printf("%d\n",maximumAverageSubArray(x,7,3));
    return 0;
}

```

Ques10.MaximumSubArrayOfLengthK

```

#include <stdio.h>
#include<limits.h>
struct record
{
    int si;
    int ei;
    int maxSum;
};
struct record maxSubArrayOfLengthK(int x[],int size,int lengthOfSubArray)
{
    int i;
    int sum=0;
    for(i=0;i<lengthOfSubArray;i++) sum+=x[i];
    int maxSum=sum;
    struct record r;
    for(i=lengthOfSubArray;i<size;i++)
    {
        sum=sum+x[i]-x[i-lengthOfSubArray];
        if(sum>maxSum)
        {

```

```

        maxSum=sum;
        r.si=i-lengthOfSubArray+1;
        r.ei=i;
    }
}
r.maxSum=maxSum;
return r;
}

int main() {
    int x[]={11,-8,-16,-7,24,-2,3};
    int lengthOfSubArray=3;
    struct record r=maxSubArrayOfLengthK(x,7,3);
    printf("StartIndex: %d\n",r.si);
    printf("EndIndex: %d\n",r.ei);
    printf("MaxSum :%d\n",r.maxSum);
    return 0;
}

```

Ques 11.MaxSizeOfSubArray

```

#include <stdio.h>
#include<limits.h>

int maxSizeOfSubArray(int x[],int size)
{
    int maxSum,currentSum,count;
    maxSum=0;
    currentSum=0;
    count=0;
    for(int i=0;i<size;i++)
    {
        currentSum=currentSum+x[i];
        if(currentSum<0)
        {
            count=0;
            currentSum=0;
        }
        if(currentSum>maxSum)
        {
            count++;
            currentSum=maxSum;
        }
    }
    return count;
}

int main() {
    int x[]={11,-8,-16,-7,-24,-2,3};
    printf("Size : %d\n",maxSizeOfSubArray(x,7));
    return 0;
}

```



```
}
```

Ques12. maximumSizeOfSubArray

```
#include <stdio.h>
```

```
#include<limits.h>
```

```
int maxSizeOfSubArray(int x[],int size)
```

```
{
    int maxSum,currentSum,count;
    maxSum=0;
    currentSum=0;
    count=0;
    for(int i=0;i<size;i++)
    {
        currentSum=currentSum+x[i];
        if(currentSum<0)
        {
            count=0;
            currentSum=0;
        }
        if(currentSum>maxSum)
        {
            count++;
            currentSum=maxSum;
        }
    }
    return count;
}
```

```
int main() {
    int x[]={11,-8,-16,-7,-24,-2,3};
    printf("Size : %d\n",maxSizeOfSubArray(x,7));
    return 0;
}
```

Ques13.SubArrayWithGivenSum

```
#include <stdio.h>
```

```
#include<limits.h>
```

```
void subArrayWithGivenSum(int x[],int size,int givenSum)
```

```
{
    int left,right,currentSum;
    left=right=currentSum=0;
    for(int i=0;i<size;i++)
    {
        if(currentSum==givenSum) break;
        if(currentSum<givenSum)
        {
```

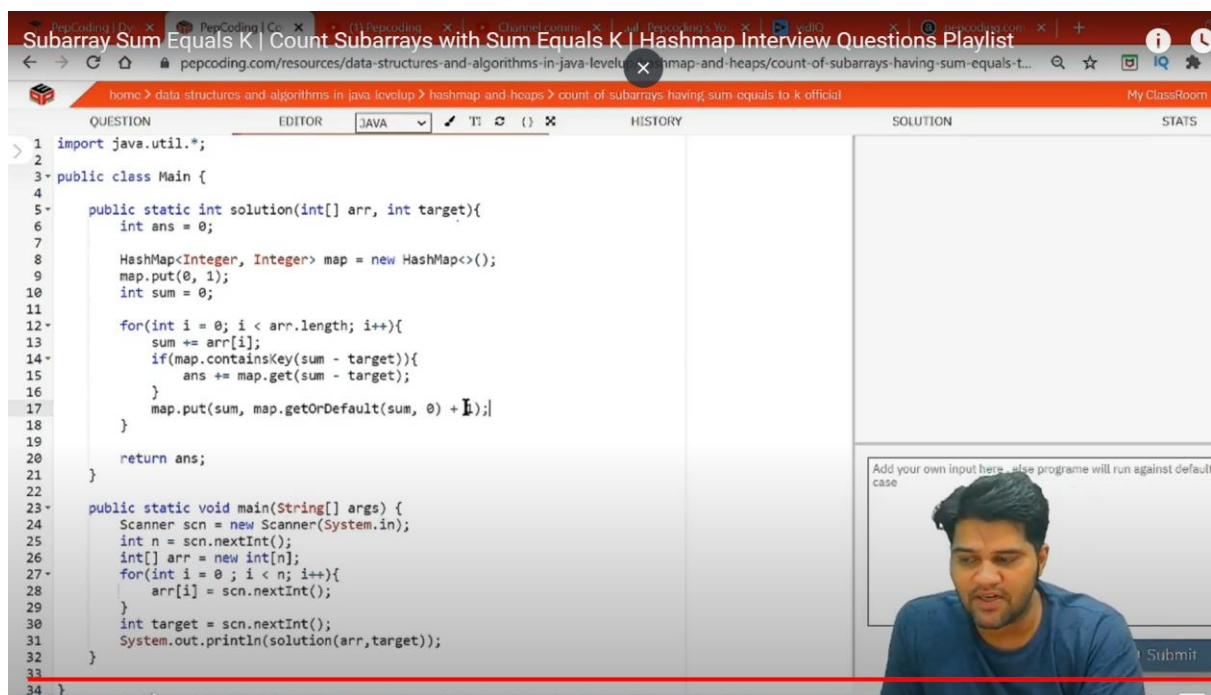
```

        currentSum=currentSum+x[right];
        right++;
    }
    else
    {
        currentSum=currentSum-x[left];
        left++;
    }
}
if((right-left+1)>size)
{
    printf("Not Found\n");
}
else
{
    printf("StartIndex :%d,endIndex :%d\n",left,right-1);
}
return ;
}

int main() {
    int x[]={1,4,-2,5,10,5};
    int size=33;
    int givenSum=3;
    subArrayWithGivenSum(x,6,givenSum);
    return 0;
}

```

Ques14. SubArray Sum Equals k



The screenshot shows a web browser with the URL `pepcoding.com/resources/data-structures-and-algorithms-in-java-levelup/hashmap-and-heaps/count-of-subarrays-having-sum-equals-t...`. The page title is "Subarray Sum Equals K | Count Subarrays with Sum Equals K | Hashmap Interview Questions Playlist". The main content is a code editor with the following Java code:

```

1 import java.util.*;
2
3 public class Main {
4
5     public static int solution(int[] arr, int target){
6         int ans = 0;
7
8         HashMap<Integer, Integer> map = new HashMap<>();
9         map.put(0, 1);
10        int sum = 0;
11
12        for(int i = 0; i < arr.length; i++){
13            sum += arr[i];
14            if(map.containsKey(sum - target)){
15                ans += map.get(sum - target);
16            }
17            map.put(sum, map.getOrDefault(sum, 0) + 1);
18        }
19
20        return ans;
21    }
22
23    public static void main(String[] args) {
24        Scanner scn = new Scanner(System.in);
25        int n = scn.nextInt();
26        int[] arr = new int[n];
27        for(int i = 0; i < n; i++){
28            arr[i] = scn.nextInt();
29        }
30        int target = scn.nextInt();
31        System.out.println(solution(arr,target));
32    }
33 }
34

```

On the right side of the code editor, there is a "SOLUTION" tab and a "STATS" tab. Below the code, there is a text input field with the placeholder "Add your own input here, else program will run against default case" and a "Submit" button. A man is visible in the bottom right corner of the video frame.

```
import java.util.*;
public class psp{

    public static void main(String []args){
        int x[]={3,9,-2,4,1,-7,2,6,-5,8,-3,-7,6,2,1};
        int sum=5;
        int i=countOfSubArrayWithGivenSum(x,sum);
        System.out.println(i);
    }
    public static int countOfSubArrayWithGivenSum(int x[],int givenSum)
    {
        int ans=0;
        HashMap<Integer,Integer> map=new HashMap<>();
        map.put(0,1);
        int sum=0;
        for(int i=0;i<x.length;i++)
        {
            sum=sum+x[i];
            if(map.containsKey(sum-givenSum))
            {
                ans=ans+map.get(sum-givenSum);
            }
            map.put(sum,map.getOrDefault(sum,0)+1);
        }
        return ans;
    }
}
```
