1. Given a sorted array A (sorted in ascending order), having N integers, find if there exists any pair of elements (A[i], A[j]) such that their sum is equal to X.

**For Example**: A[] = {10, 20, 35, 50, 75, 80} and the value of X = 110

Solution:

**publicstaticint**isPairSum(**int**A[], **int**N, **int**X)

    {

        // represents first pointer

**int**i = 0;

        // represents second pointer

**int**j = N - 1;

**while**(i < j) {

            // If we find a pair

**if**(A[i] + A[j] == X)

**return**1;

            // If sum of elements at current

            // pointers is less, we move towards

            // higher values by doing i++

**elseif**(A[i] + A[j] < X)

                i++;

            // If sum of elements at current

            // pointers is more, we move towards

            // lower values by doing j--

**else**

                j--;

        }// while loop closed

**return**0;

    }

TC: O(n)

SC: O(1)



1. Given an array of distinct elements. The task is to find triplets in the array whose sum is zero.

Examples: Input: arr[] = {0, -1, 2, -3, 1}



Output: (0 -1 1), (2 -3 1)



Explanation: The triplets with zero sum are 0 + -1 + 1 = 0 and 2 + -3 + 1 = 0

Input: arr[] = {1, -2, 1, 0, 5}

Output: 1 -2 1

Explanation: The triplets with zero sum is 1 + -2 + 1 = 0

**Solution:**

// function to print triplets with 0 sum

static void findTriplets(int arr[], int n)

{

boolean found = false;

// sort array elements

Arrays.sort(arr);



for (int i = 0; i < n - 1; i++) {

// initialize left and right

int l = i + 1;

int r = n - 1;

int x = arr[i];

while (l < r) {

if (x + arr[l] + arr[r] == 0) {



// print elements if it's sum is zero

System.out.print(x + " ");

System.out.print(arr[l] + " ");

System.out.println(arr[r] + " ");

l++;

r--;

found = true;

}// if closed

// If sum of three elements is less

// than zero then increment in left

else if (x + arr[l] + arr[r] < 0)

l++;

// if sum is greater than zero then

// decrement in right side

else

r--;

}// while loop closed

}// for loop closed

if (found == false)

System.out.println(" No Triplet Found");

}// function closed

TC: O(n\*n)

SC: O(1)

1. Given a sorted array, remove all duplicates and return the length of the modified array solution.

Examples: nums = {1, 1, 2, 2, 2, 3, 4, 4}; Output: 4

**Solution:**

public int removeDuplicates(int[] nums) {

if (nums == null || nums.length == 0) {

return 0;

}

int len = 1;

for (int i = 1; i < nums.length; i++) {

if (nums[i] != nums[i-1]) {

nums[len] = nums[i];

len++;

}

}

return len;

}

TC: O(n)

SC: O(1)



1. Given an array of integers, move all the zeros to the end of the array while maintaining the relative order of the other elements.

Input: Arr[]={10,23,0,53,0,21,0}

Output: Arr[]={10,23,53,21,0,0,0}

**Solution:**

public void moveZeroes(int[] nums) {

if (nums == null || nums.length == 0) {

return;

}

int i = 0;

for (int num : nums) {

if (num != 0) {

nums[i++] = num;

}

}

while (i< nums.length) {

nums[i++] = 0;

}

}

TC: O(n)

SC: O(1)

1. Given an array of integers, you have to find three numbers such that the sum of two elements equals the third element.



Input : {5, 32, 1, 7, 10, 50, 19, 21, 2}

Output : 21, 2, 19

**Solution:**

**staticvoid**findTriplet(**int**arr[], **int**n)

    {

        // sort the array

        Arrays.sort(arr);

        // for every element in arr

        // check if a pair exist(in array) whose

        // sum is equal to arr element

**for**(**int**i = n - 1; i >0; i--) {

**int**j = 0;

**int**k = i - 1;

**while**(j < k) {

**if**(arr[i] == arr[j] + arr[k]) {

                    // pair found

                    System.out.println("numbers are "+ arr[i] + " "

                                       + arr[j] + " "+ arr[k]);

**return**;

                }

**elseif**(arr[i] > arr[j] + arr[k])

                    j += 1;//j++

**else**

                    k -= 1;//k--

            }

        }

        // no such triplet is found in array

        System.out.println("No such triplet exists");

    }

TC: O(n\*n)

SC: O(1)