

# 3 Sum

Problem	Submissions	Leaderboard	Discussions
---------	-------------	-------------	-------------

Take an integer array `arr` as input and print all the triplets `[arr[i], arr[j], arr[k]]` such that  $i \neq j$ ,  $i \neq k$ , and  $j \neq k$ , and  $(arr[i] + arr[j] + arr[k] == 0)$ .

Notice that the solution set must not contain duplicate triplets.

Handwritten diagram showing an array: `6`  
`-2 0 2 4 -2 -8`  
Indices `0 1 2 3 4 5` are written below. Arrows point from the index `6` to the elements `-2`, `0`, `2`, and `4`. A box encloses the elements `-2`, `0`, and `2`.

Handwritten diagram showing two identical triplets: `-2 0 2`. A large 'X' is drawn to the left of the first triplet, and an arrow points from the second triplet to the number `0`.

Handwritten diagram showing a set of elements: `-2 -2 4`. Checkmarks are placed above each element. An arrow points from the set to the number `0`. Above the set, the text `-arr = 0` is written.

## Sample Output 0

Handwritten diagram showing a set of elements: `-2 -2 4` and `-2 0 2`. Arrows point from the sets to the number `0`.

6

-2 0 2 4 -2 -8

→ sort

-8  
0-2  
1  
↑  
i-2  
2  
l0  
32  
44  
5  
r

2

$$t = 0$$

$$a + b + c = 0$$

$$-2 \quad | \quad b + c = 2$$

$$nTr = 2$$

$$S = 2$$

-2 0 2

2 -2 4

$$-8$$

$$\begin{bmatrix} -2 \end{bmatrix}$$

$$\begin{matrix} -2 \\ \textcircled{i} \\ 1 \end{matrix}$$

$$\begin{bmatrix} 0 \\ l \end{bmatrix}$$

$$2$$

$$\begin{bmatrix} 4 \end{bmatrix}$$

$$\rightarrow l = i+1$$

$$\underline{r} = n-1$$

```

for(int i = 0; i < n; i++){
    if(i != 0 && A[i] == A[i-1]){
        continue;
    }

    int l = i+1;
    int r = n-1;
    int nTr = 0 - (A[i]);

    while(l < r){
        int s = A[l] + A[r];
        if(s == nTr){
            System.out.println(A[i] + " " + A[l] + " " + A[r]);
            l++;
            r--;
            while(l < r && A[l] == A[l-1]){
                l++;
            }
            while(l < r && A[r] == A[r+1]){
                r--;
            }

        }else if(s > nTr){
            r--;
        }else{ // s < nTr
            l++;
        }
    }
}

```

4  
9  
8

```

10
-8 -2 -2 -2 -2 0 2 4 4 4

```

Your Output

```

-8 4 4
-2 -2 4
-2 0 2

```

$n^2$  $n^2 + n \log n \Rightarrow O(n^2)$ 

```
1 import java.io.*;
2 import java.util.*;
3
4 public class Solution {
5     public static void main(String[] args) {
6         Scanner scn = new Scanner(System.in);
7         int n = scn.nextInt();
8         int [] A = new int[n];
9         for(int i = 0; i < n; i++){
10             A[i] = scn.nextInt();
11         }
12         Arrays.sort(A);
13
14         for(int i = 0; i < n; i++){
15             if(i != 0 && A[i] == A[i-1]){
16                 continue;
17             }
18
19             int l = i+1;
20             int r = n-1;
21             int nTr = 0 - (A[i]);
22
23             while(l < r){
24                 int s = A[l] + A[r];
25                 if(s == nTr){
26                     System.out.println(A[i] + " " + A[l] + " " + A[r]);
27                     l++;
28                 }
29             }
30         }
31     }
32 }
```

```
28         r--;
29         while(l < r && A[l] == A[l-1]){
30             l++;
31         }
32         while(l < r && A[r] == A[r+1]){
33             r--;
34         }
35
36         }else if(s > nTr){
37             r--;
38         }else{ // s < nTr
39             l++;
40         }
41     }
42 }
43 }
44 }
45 }
```

# Four Sum

[Problem](#)[Submissions](#)[Leaderboard](#)[Discussions](#)

The given array is not sorted. The given array may or may not contain duplicate elements. Then take the target as an integer input. Print all the **unique quadruple** whose sum is equal **target**.

**NOTE** all quadruple should be unique, for example :  $[6, 7, 8, 9]$ ,  $[7, 6, 8, 9]$  are considered as same quadruple. Also if the array has repeated elements then return only unique quadruple, for eg : if array is  $arr = [3, 3, 5, 5, 1, 1, 2, 2]$ , and the  $target = 11$ , then result will have only one quadruple, i.e.  $[1, 2, 3, 5]$ . The result should be sorted in increasing order and also the quadruple.

Given a binary string `s`, return the number of **non-empty** substrings that have the same number of 0's and 1's, and all the 0's and all the 1's in these substrings are grouped consecutively. Substrings that occur multiple times are counted the number of times they occur.

Sample Input 0

```
00110011
```

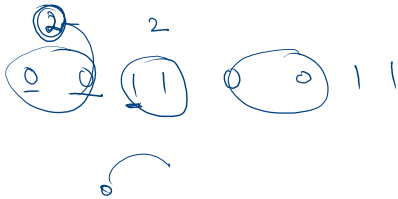
Sample Output 0

```
6
```

$$\begin{array}{ccc} 0 & 1 & 1 \\ \hline & & \end{array} \rightarrow 1$$
  
$$\begin{array}{ccc} & 0 & 0 & 1 \\ & \hline & & \end{array} \rightarrow 1$$

$$\begin{array}{cccc} 0 & 0 & 1 & 1 \\ \hline & & & \end{array} \rightarrow 2$$

$$000 \quad 111 \quad 1 \rightarrow 3$$



$$\begin{array}{ccc} 000 & 11 \\ \hline 3 & 2 \end{array} \quad \min(2, 3) = 2$$
  
$$2 + 2 + 2 = 6$$

$$\begin{array}{ccc} 0 & 0 & 1 & 1 \\ \hline 2 & 2 \end{array} \quad \begin{array}{ccc} 0 & 0 & 1 & 1 \\ \hline 2 & 2 \end{array}$$

