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 != j, i != k, and j != k`, and `arr[i] + arr[j] + arr[k] == 0.`

Notice that the solution set must not contain duplicate triplets.

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

tar = 0

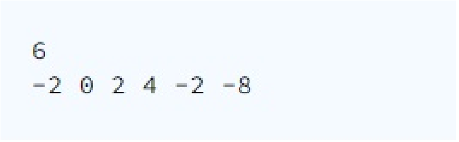
$-2 \quad -2 \quad 4 \rightarrow 0$

$-2 \quad 0 \quad 2 \rightarrow 0$

$-2 \quad 0 \quad 2$

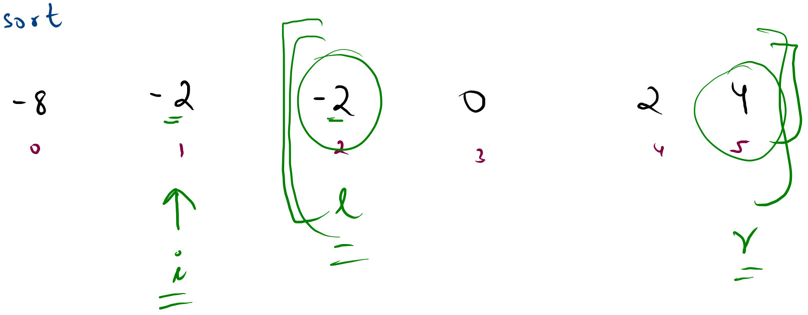
Sample Output 0

```
-2 -2 4
-2 0 2
```



```
6
-2 0 2 4 -2 -8
```

→ sort



t = 0

a + b + c = 0

-2 $\boxed{b + c = 2}$

S = 2

nTr = 2

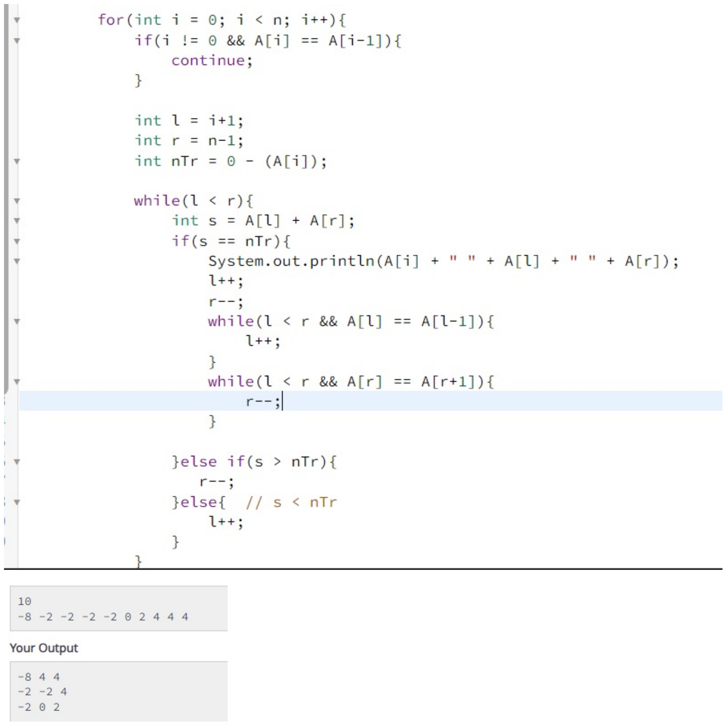
$\boxed{-2 \quad 0 \quad 2}$

2 -2 4

$$\underset{i}{-8} \qquad \left[\; -2 \right. \qquad \underset{\textcircled{i}}{-2} \qquad \underset{l}{\left[\; 0 \quad 2 \quad 4 \;\right]}$$

$$\rightarrow l = i+1$$

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



```
        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++;
                }
            }
```

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

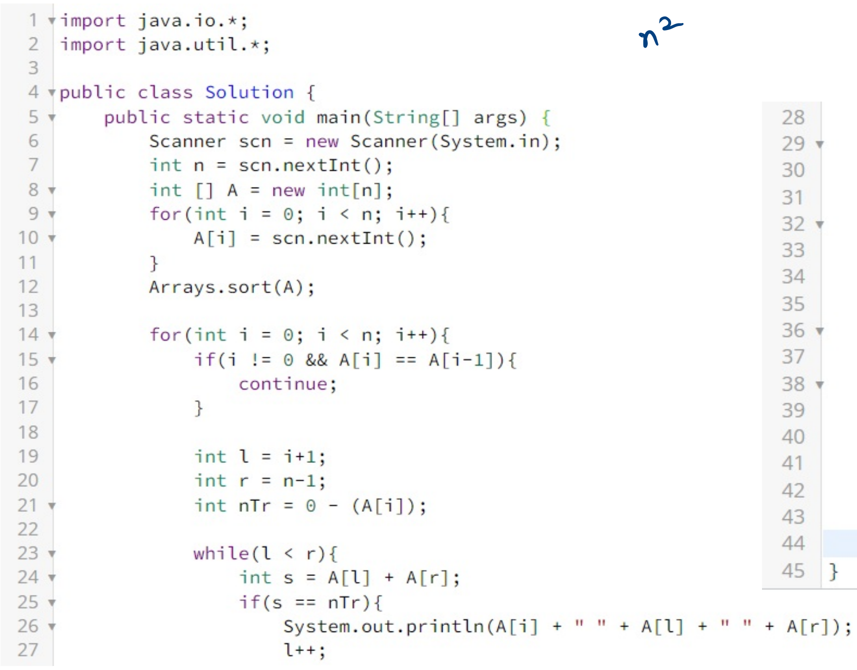
Your Output

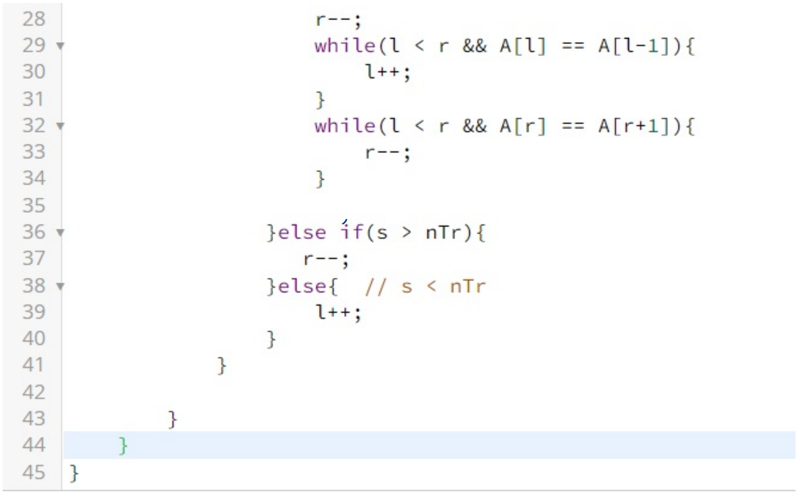
```
-8 4 4
-2 -2 4
-2 0 2
```

Y
9
γ

n^2

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





```java
import java.io.*;
import java.util.*;

public class Solution {
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        int [] A = new int[n];
        for(int i = 0; i < n; i++){
            A[i] = scn.nextInt();
        }
        Arrays.sort(A);

        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++;
                }
            }

        }
    }
}
```

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

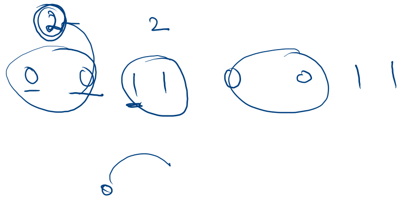
```
6
```

$\underline{0\ 1\ 1} \rightarrow 1$

$0\ 0\ 1 \rightarrow 1$

$\underline{0\ 0\ 1\ 1\ 1} \rightarrow 2$

$0\ 0\ 0\ 1\ 1\ 1\ 1 \rightarrow 3$



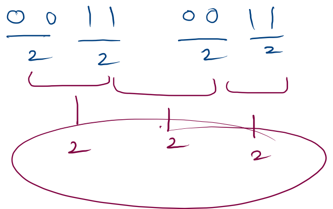
$\underbrace{0\ 0\ 0}_{3}\ \underbrace{1\ 1}_{2}$

$\underline{\min(2,3)}$

$= 2$

$2 + 2 + 2 = \boxed{6}$

$0\ 0\ 1\ 1 \quad \underset{2}{\underline{0\ 0}}\ \underset{2}{\underline{1\ 1}}$



0 0
2
1 1
2
0 0
2
1 1
2
2
2
2