

# Triplet Sum Close to Target (medium)

We'll cover the following



- Problem Statement
- Try it yourself
- Solution
  - Code
  - Time complexity
  - Space complexity

## Problem Statement #

Given an array of unsorted numbers and a target number, find a **triplet in the array whose sum is as close to the target number as possible**, return the sum of the triplet. If there are more than one such triplet, return the sum of the triplet with the smallest sum.

### Example 1:

Input: [-2, 0, 1, 2], target=2

Output: 1

Explanation: The triplet [-2, 1, 2] has the closest sum to the target.

### Example 2:

Input: [-3, -1, 1, 2], target=1

Output: 0

Explanation: The triplet [-3, 1, 2] has the closest sum to the target.



### Example 3:

Input: [1, 0, 1, 1], target=100

Output: 3

Explanation: The triplet [1, 1, 1] has the closest sum to the target.

## Try it yourself #

Try solving this question here:

Java

Python3

JS

C++

```
6      if (min==0){
7          return ans.reduce(reducer);
8      }
9  }
10     return ans.reduce(reducer);
11 };
12 //
13 const helper = (arr, left_value, mid, target_sum, min) => {
14     let right = arr.length-1;
15     while (mid<right){
16         const curr_value = left_value+arr[mid]+arr[right];
17         curr_min = curr_value-target_sum;
18         if (min>Math.abs(curr_min)){
19             min = Math.abs(curr_min);
20             ans = [left_value, arr[mid], arr[right]]
21         }
22         if (curr_min==0){
23             return [min, ans];
24         } else if (curr_min>0){
25             right--;
26         } else{
27             mid++;
28         }
29     }
30 }
```



```

29     }
30     return [min,ans];
31 }
32
33 const reducer = (accumulator, currentValue) => a

```



Show Results

Show Console



2.53s

3 of 3 Tests Passed

Result	Input	Expected Output	Actual Output	Real
✓	triplet_sum_close_to_target([-2,0,1,2], ...	1	1	Success
✓	triplet_sum_close_to_target([-3,-1,1,2], ...	0	0	Success
✓	triplet_sum_close_to_target([1,0,1,1], 1 ...	3	3	Success

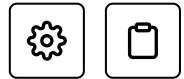
## Solution #

This problem follows the **Two Pointers** pattern and is quite similar to Triplet Sum to Zero

(<https://www.educative.io/collection/page/5668639101419520/5671464854355968/5679549973004288/>).

We can follow a similar approach to iterate through the array, taking one number at a time. At every step, we will save the difference between the triplet and the target number, so that in the end, we can return the triplet with the closest sum.

## Code #



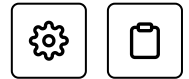
Here is what our algorithm will look like:

Java	Python3	C++	JS JS
<pre> 1 function triplet_sum_close_to_target(arr, targetSum) { 2   arr.sort((a, b) =&gt; a - b); 3   let smallest_difference = Infinity; 4   for (let i = 0; i &lt; arr.length - 2; i++) { 5     let left = i + 1, 6       right = arr.length - 1; 7     while (left &lt; right) { 8       const target_diff = targetSum - arr[i] - arr[right]; 9       if (target_diff === 0) { // we've found a triplet that sums up to the target 10        return targetSum - target_diff; // return the sum of the triplet 11      } 12 13      if (Math.abs(target_diff) &lt; Math.abs(smallest_difference)) { 14        smallest_difference = target_diff; // save the smallest difference 15      } 16      // the second part of the following 'if' is to handle the case where the target_diff is negative 17      if (Math.abs(target_diff) &lt; Math.abs(smallest_difference)) { 18        (Math.abs(target_diff) === Math.abs(smallest_difference)) ? smallest_difference = target_diff : smallest_difference = target_diff; // save the smallest difference 19      } 20    } 21 22    if (target_diff &gt; 0) { 23      left += 1; // we need a triplet with a larger sum 24    } else { 25      right -= 1; // we need a triplet with a smaller sum 26    } 27  } 28 } </pre>			

## Time complexity #

Sorting the array will take  $O(N * \log N)$ . Overall, the function will take  $O(N * \log N + N^2)$ , which is asymptotically equivalent to  $O(N^2)$ .

## Space complexity #



The above algorithm's space complexity will be  $O(N)$ , which is required for sorting.

[← Back](#)[Next →](#)[Triplet Sum to Zero \(medium\)](#)[Triplets with Smaller Sum \(medium\)](#)[Mark as Completed](#)[Report an Issue](#)[Ask a Question](#)

([https://discuss.educative.io/tag/triplet-sum-close-to-target-medium\\_\\_pattern-two-pointers\\_\\_grokking-the-coding-interview-patterns-for-coding-questions](https://discuss.educative.io/tag/triplet-sum-close-to-target-medium__pattern-two-pointers__grokking-the-coding-interview-patterns-for-coding-questions))