Back To Course Home

Grokking the Coding Interview: Patterns for Coding Questions

7% completed

Search Course

interview/N069QhPLKNv)

Problem Challenge 2 (/courses/grokking-the-coding-interview/YQ8N2OZq0VM)

Solution Review: Problem Challenge 2 (/courses/grokking-the-coding-interview/xl2g3vxrMq3)

Problem Challenge 3 (/courses/grokking-the-coding-interview/3wDJAYG2pAR)

Solution Review: Problem Challenge 3 (/courses/grokking-the-coding-interview/xoyL4q6ApNE)

Problem Challenge 4 (/courses/grokking-the-coding-

Triplet Sum Close to Target (medium)

We'll cover the following

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++

```
import java.util.*;

class TripletSumCloseToTarget {

  public static int searchTriplet(int[] arr, int ta
    // TODO: Write your code here
    Arrays.sort(arr);
    int n = arr.length,ans = Integer.MAX_VALUE;
    for(int i=0;i<arr.length-2;i++)
    {
      int start = i+1,end = n-1;
      while(start<end)      {
```

```
      while(start<end)      {
        int diff = targetSum-arr[i]-arr[start]-arr
        if(diff==0){
          return  targetSum-diff;
        }
        ans = Math.min(ans, Math.abs(diff));
        if(diff>0){
          start++;
        }
        else{
          end--;
        }
      }
    }
    return targetSum-ans;
  }
}
```

Test | Save * | Reset

Show Results | Show Console

3 of 3 Tests Passed

Result	Input	Expected Output	Actual Output	Reason
✓	searchTriplet([-2, 0, 1, 2], 2)	1	1	Succeeded
✓	searchTriplet([-3, -1, 1, 2], 1)	0	0	Succeeded
✓	searchTriplet([1, 0, 1, 1], 100)	3	3	Succeeded

3.883s

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

```
import java.util.*;

class TripletSumCloseToTarget {

  public static int searchTriplet(int[] arr, int ta
    if (arr == null || arr.length < 3)
      throw new IllegalArgumentException();

    Arrays.sort(arr);
    int smallestDifference = Integer.MAX_VALUE;
    for (int i = 0; i < arr.length - 2; i++) {
      int left = i + 1, right = arr.length - 1;
      while (left < right) {
        // comparing the sum of three numbers to th
        // so, we will try to find a target differe
        int targetDiff = targetSum - arr[i] - arr[l
        if (targetDiff == 0) //  we've found a trip
          return targetSum - targetDiff; // return

```

```
        // the second part of the above 'if' is to
        if (Math.abs(targetDiff) < Math.abs(smalles
            || (Math.abs(targetDiff) == Math.abs(sm
          smallestDifference = targetDiff; // save

        if (targetDiff > 0)
          left++; // we need a triplet with a bigge
        else
          right--; // we need a triplet with a smal
      }
    }
    return targetSum - smallestDifference;
```

Run Save Reset

Time complexity

Sorting the array will take $O(N * logN)$. Overall `searchTriplet()` will take $O(N * logN + N^2)$, which is asymptotically equivalent to $O(N^2)$.

Space complexity

The space complexity of the above algorithm will be $O(N)$ which is required for sorting.

Back

Triplet Sum to Zero (medium)

Next

Triplets with Smaller Sum (medium)

Mark as Completed

7% completed, meet the criteria and claim your course certificate!

Claim Certificate

Grokking the Coding Interview: Patterns for Coding Questions