Back To Course Home

Grokking the Coding Interview: Patterns for Coding Questions

6% completed

Search Course

interview/N8vB7OVYo2D)

Solution Review: Problem Challenge 1 (/courses/grokking-the-coding-interview/N0o9QnPLKNv)

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

Triplet Sum to Zero (medium)

We'll cover the following

Problem Statement

Given an array of unsorted numbers, find all **unique triplets in it that add up to zero**.

Example 1:

```
Input: [-3, 0, 1, 2, -1, 1, -2]
Output: [-3, 1, 2], [-2, 0, 2], [-2, 1, 1], [-1, 0, 1]
Explanation: There are four unique triplets whose sum is equal to zero.
```

Example 2:

Back To Course Home

Grokking the Coding Interview: Patterns for Coding Questions

6% completed

Search Course

interview/N8vB7OVYo2D)

Solution Review: Problem Challenge 1 (/courses/grokking-the-coding-interview/N0o9QnPLKNv)

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

```
Input: [-5, 2, -1, -2, 3]
Output: [[-5, 2, 3], [-2, -1, 3]]
Explanation: There are two unique triplets whose sum is equal to zero.
```

Try it yourself

Try solving this question here:

Java | Python3 | JS | C++

```
      int sum = arr[left] + arr[right];
      if(sum == target){
        triplets.add(Arrays.asList(-target,arr[left
        left++;
        right--;
        while(left<right&&arr[left]==arr[left-1]){
          left++;
        }
        while(left<right&&arr[right]==arr[right+1])
          right--;
        }
      }
      else if(sum>target){
        right--;
      }
      else{
        left++;
      }
    }
  }
  public static List<List<Integer>> searchTriplets(
    List<List<Integer>> triplets = new ArrayList<>(
    Arrays.sort(arr);
    for(int i=0;i<arr.length-2;i++){
      if(i>0&&arr[i-1]==arr[i]){
        continue;
      }
      searchPair(arr,triplets,-arr[i],i+1);
```

Back To Course Home

Grokking the Coding Interview: Patterns for Coding Questions

6% completed

Search Course

interview/N8vB7OVYo2D)

Solution Review: Problem Challenge 1 (/courses/grokking-the-coding-interview/N0o9QnPLKNv)

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

```
34      }
35      // TODO: Write your code here
```

Test | Save * | Reset

Show Results | Show Console

2 of 2 Tests Passed

Result	Input	Expected Output	Actual Output	Reason
✓	searchTriplets([-3, 0, 1, 2, -1, 1, -2])	[[-3, 1, 2], [-2, 1, 1], [-2, 0, 2], [-1 ⋯	[[-3, 1, 2], [-2, 0, 2], [-2, 1, 1], [-1 ⋯	Succeeded
✓	searchTriplets([-5, 2, -1, -2, 3])	[[-2, -1, 3], [-5, 2, 3]]	[[-5, 2, 3], [-2, -1, 3]]	Succeeded

10.960s

Solution

This problem follows the **Two Pointers** pattern and shares similarities with Pair with Target Sum (https://www.educative.io/collection/page/5668639101419520/5671464854355968/6618310940557312/). A couple of differences are that the input array is not sorted and instead of a pair we need to find triplets with a target sum of zero.

To follow a similar approach, first, we will sort the array and then iterate through it taking one number at a time. Let's say during our iteration we are at number 'X', so we need to find 'Y' and 'Z' such that $X + Y + Z == 0$. At this stage, our problem

translates into finding a pair whose sum is equal to "$-X$" (as from the above equation $Y + Z == -X$).

Another difference from Pair with Target Sum (https://www.educative.io/collection/page/5668639101419520/5671464854355968/6618310940557312/) is that we need to find all the unique triplets. To handle this, we have to skip any duplicate number. Since we will be sorting the array, so all the duplicate numbers will be next to each other and are easier to skip.

Code

Here is what our algorithm will look like:

Java | Python3 | C++ | JS

```
import java.util.*;

class TripletSumToZero {

  public static List<List<Integer>> searchTriplets(
    Arrays.sort(arr);
    List<List<Integer>> triplets = new ArrayList<>(
    for (int i = 0; i < arr.length - 2; i++) {
      if (i > 0 && arr[i] == arr[i - 1]) // skip sa
        continue;
      searchPair(arr, -arr[i], i + 1, triplets);
    }

    return triplets;
  }

  private static void searchPair(int[] arr, int tar
    int right = arr.length - 1;
    while (left < right) {
      int currentSum = arr[left] + arr[right];
      if (currentSum == targetSum) { // found the t
        triplets.add(Arrays.asList(-targetSum, arr[
```

Back To Course Home

Grokking the Coding Interview: Patterns for Coding Questions

6% completed

Search Course

interview/N8vB7OVYo2D)

Solution Review: Problem Challenge 1 (/courses/grokking-the-coding-interview/N0o9QnPLKNv)

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

```
23          left++;
24          right--;
25          while (left < right && arr[left] == arr[le
26            left++; // skip same element to avoid du
27          while (left < right && arr[right] == arr[r
28            right--; // skip same element to avoid d
29        } else if (targetSum > currentSum)
30          left++; // we need a pair with a bigger su
31        else
```

Run Save Reset

Time complexity

Sorting the array will take $O(N * logN)$. The `searchPair()` function will take $O(N)$. As we are calling `searchPair()` for every number in the input array, this means that overall `searchTriplets()` will take $O(N * logN + N^2)$, which is asymptotically equivalent to $O(N^2)$.

Space complexity

Ignoring the space required for the output array, the space complexity of the above algorithm will be $O(N)$ which is required for sorting.

Back

Squaring a Sorted Array (easy)

Next

Triplet Sum Close to Target (medium)

Mark as Completed

6% completed, meet the criteria and claim your course certificate!

Back To Course Home

Grokking the Coding Interview: Patterns for Coding Questions

6% completed

Search Course

Report an Issue

Ask a Question (https://discuss.educative.io/tag/triplet-sum-to-zero-medium__pattern-two-pointers__grokking-the-coding-interview-patterns-for-coding-questions)