

Triplet Sum to Zero (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, 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:





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++
--	---	--	---

```
1 def search_triplets(arr):
2     triplets = []
3     # TODO: Write your code here
4     return triplets
5
6
```

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

```

3   triplets = []
4   for i in range(len(arr)):
5       if i > 0 and arr[i] == arr[i-1]: # skip same element to avoid duplicate triplets
6           continue
7       search_pair(arr, -arr[i], i+1, triplets)
8
9   return triplets
10
11
12 def search_pair(arr, target_sum, left, triplets):
13     right = len(arr) - 1
14     while(left < right):
15         current_sum = arr[left] + arr[right]
16         if current_sum == target_sum: # found the triplet
17             triplets.append([-target_sum, arr[left], arr[right]])
18             left += 1
19             right -= 1
20             while left < right and arr[left] == arr[left - 1]:
21                 left += 1 # skip same element to avoid duplicate triplets
22             while left < right and arr[right] == arr[right + 1]:
23                 right -= 1 # skip same element to avoid duplicate triplets
24         elif target_sum > current_sum:
25             left += 1 # we need a pair with a bigger sum
26         else:
27             right -= 1 # we need a pair with a smaller sum
28
29
30 def main():

```

Sorting the array will take $O(N * \log N)$. 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 $O(N * \log N + N^2)$, 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](#)
[Next →](#)
[Squaring a Sorted Array \(easy\)](#)
[Triplet Sum Close to Target \(medium\)](#)
☒ **Mark as Completed**

[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)