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:

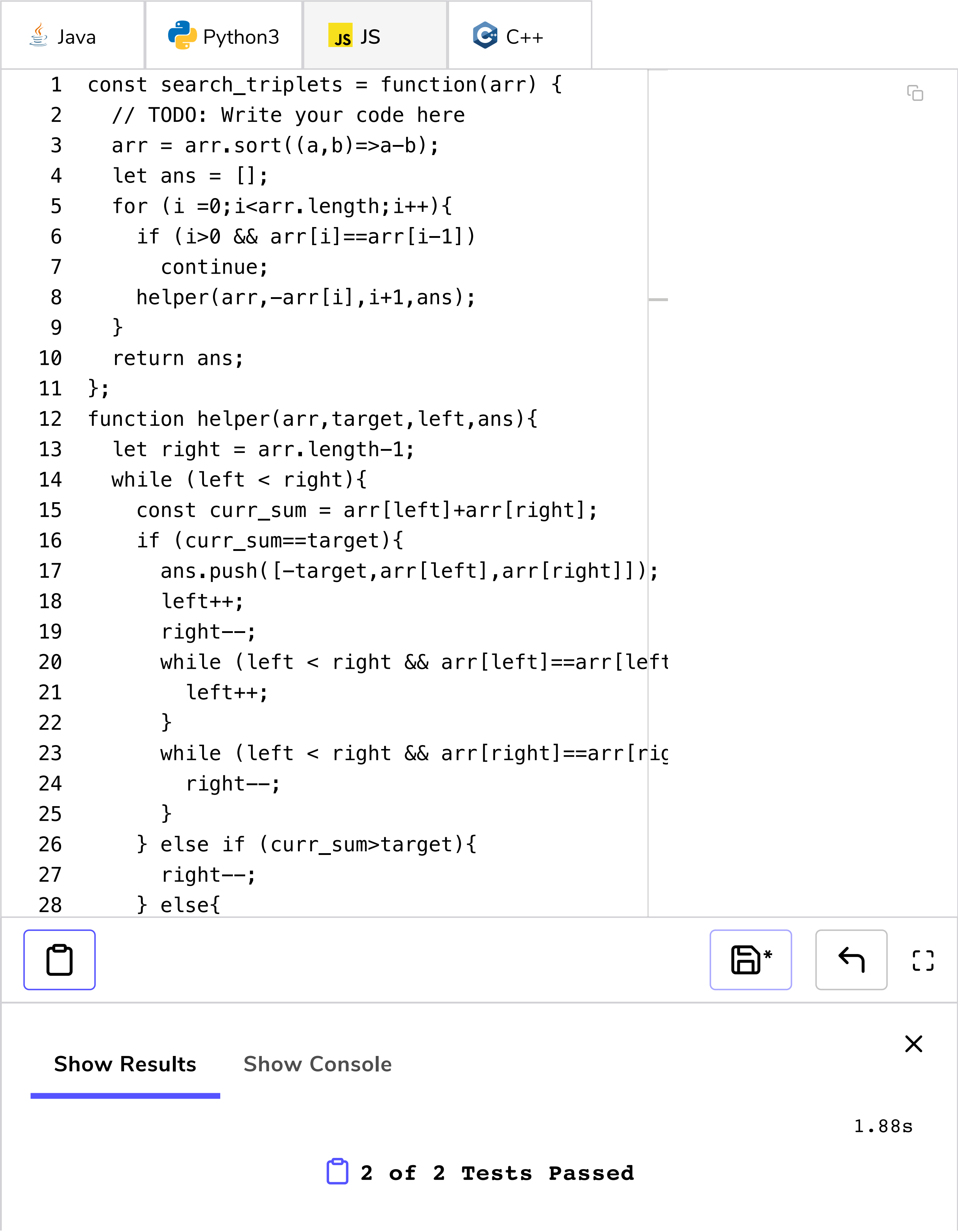
```
Input: [-3, 0, 1, 2, -1, 1, -2]
Output: [-3, 1, 2], [-2, 0, 2], [-2, 1, 1], [-1, 0, 1]
Explanation: There are four unique triplets whose sum is equal t
o zero.
```

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 z
ero.
```

Try it yourself

Try solving this question here:



```
const search_triplets = function(arr) {
  // TODO: Write your code here
  arr = arr.sort((a,b)=>a-b);
  let ans = [];
  for (i =0;i<arr.length;i++){
    if (i>0 && arr[i]==arr[i-1])
      continue;
    helper(arr,-arr[i],i+1,ans);
  }
  return ans;
};
function helper(arr,target,left,ans){
  let right = arr.length-1;
  while (left < right){
    const curr_sum = arr[left]+arr[right];
    if (curr_sum==target){
      ans.push([-target,arr[left],arr[right]]);
      left++;
      right--;
      while (left < right && arr[left]==arr[left
        left++;
      }
      while (left < right && arr[right]==arr[rig
        right--;
      }
    } else if (curr_sum>target){
      right--;
    } else{
```

```
1.88s

2 of 2 Tests Passed
```

Result	Input	Expected Output	Actual Output	Reas
✓	search_triplets([-3,0,1,2,-1,1,-2])	[-3,1,2] [-2,0,2] [-2,1,1] [-1,0,1]	[-3,1,2] [-2,0,2] [-2,1,1] [-1,0,1]	Succe
✓	search_triplets([-5,2,-1,-2,3])	[-5,2,3] [-2,-1,3]	[-5,2,3] [-2,-1,3]	Succe

Solution

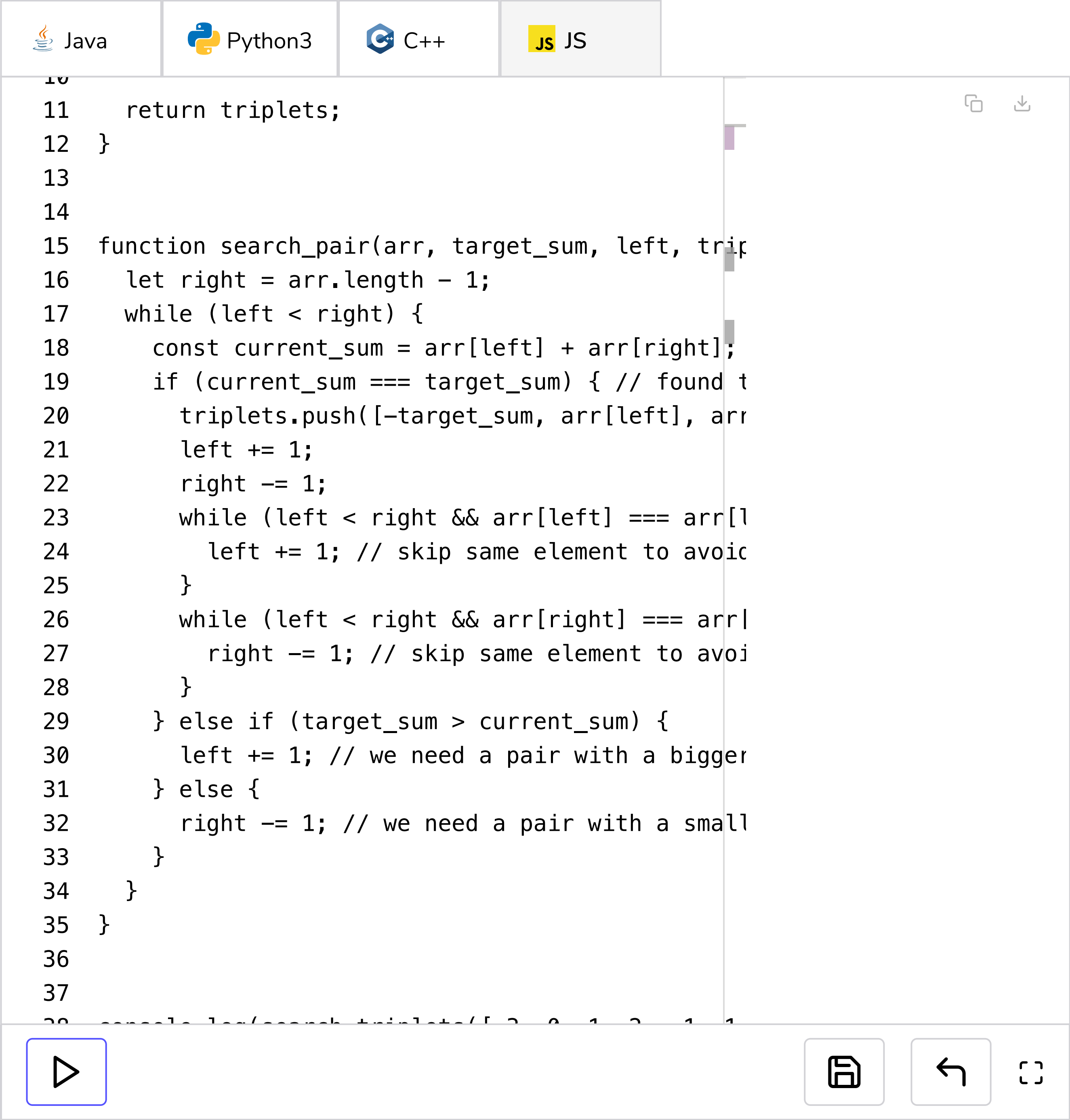
This problem follows the **Two Pointers** pattern and shares similarities with Pair with Target Sum (https://www.educative.io/collection/page/5668639101419520/5671464854355968/6618310940557312/). A couple of differences are that the input array is not sorted and instead of a pair we need to find triplets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



```
  return triplets;
}


function search_pair(arr, target_sum, left, trip
  let right = arr.length - 1;
  while (left < right) {
    const current_sum = arr[left] + arr[right];
    if (current_sum === target_sum) { // found t
      triplets.push([-target_sum, arr[left], arr
      left += 1;
      right -= 1;
      while (left < right && arr[left] === arr[l
        left += 1; // skip same element to avoic
      }
      while (left < right && arr[right] === arr[
        right -= 1; // skip same element to avoi
      }
    } else if (target_sum > current_sum) {
      left += 1; // we need a pair with a bigger
    } else {
      right -= 1; // we need a pair with a small
    }
  }
}


console.log(search_triplets([-3, 0, 1, 2, -1, 1
```

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

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)