function triplet\_with\_smaller\_sum(arr, target) {

arr.sort((a, b) => a - b);

const triplets = [];

for (i = 0; i < arr.length - 2; i++) {

search\_pair(arr, target - arr[i], i, triplets);

}

return triplets;

}

function search\_pair(arr, target\_sum, first, triplets) {

let left = first + 1,

right = arr.length - 1;

while ((left < right)) {

if (arr[left] + arr[right] < target\_sum) { // found the triplet

// since arr[right] >= arr[left], therefore, we can replace arr[right] by any number between

// left and right to get a sum less than the target sum

for (i = right; i > left; i--) {

triplets.push([arr[first], arr[left], arr[i]]);

}

left += 1;

} else {

right -= 1; // we need a pair with a smaller sum

}

}

}

console.log(triplet\_with\_smaller\_sum([-1, 0, 2, 3], 3));

console.log(triplet\_with\_smaller\_sum([-1, 4, 2, 1, 3], 5));

Java

import java.util.\*;

class TripletWithSmallerSum {

public static List<List<Integer>> searchTriplets(int[] arr, int target) {

Arrays.sort(arr);

List<List<Integer>> triplets = new ArrayList<>();

for (int i = 0; i < arr.length - 2; i++) {

searchPair(arr, target - arr[i], i, triplets);

}

return triplets;

}

private static void searchPair(int[] arr, int targetSum, int first, List<List<Integer>> triplets) {

int left = first + 1, right = arr.length - 1;

while (left < right) {

if (arr[left] + arr[right] < targetSum) { // found the triplet

// since arr[right] >= arr[left], therefore, we can replace arr[right] by any number between

// left and right to get a sum less than the target sum

for (int i = right; i > left; i--)

triplets.add(Arrays.asList(arr[first], arr[left], arr[i]));

left++;

} else {

right--; // we need a pair with a smaller sum

}

}

}

public static void main(String[] args) {

System.out.println(TripletWithSmallerSum.searchTriplets(new int[] { -1, 0, 2, 3 }, 3));

System.out.println(TripletWithSmallerSum.searchTriplets(new int[] { -1, 4, 2, 1, 3 }, 5));

}

}