JS Solution 1:

function pair\_with\_target\_sum(arr, targetSum) {

let left = 0,

right = arr.length - 1;

while (left < right) {

const currentSum = arr[left] + arr[right];

if (currentSum === targetSum) {

return [left, right];

}

if (targetSum > currentSum) {

left += 1; // we need a pair with a bigger sum

} else {

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

}

}

return [-1, -1];

}

console.log(pair\_with\_target\_sum([1, 2, 3, 4, 6], 6));

console.log(pair\_with\_target\_sum([2, 5, 9, 11], 11));

JS Solution 2:

function pair\_with\_target\_sum(arr, targetSum) {

const nums = {}; // to store numbers and their indices

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

const num = arr[i];

if (targetSum - num in nums) {

return [nums[targetSum - num], i];

}

nums[arr[i]] = i;

}

return [-1, -1];

}

console.log(pair\_with\_target\_sum([1, 2, 3, 4, 6], 6));

console.log(pair\_with\_target\_sum([2, 5, 9, 11], 11));

C++

using namespace std;

#include <iostream>

#include <vector>

class PairWithTargetSum {

public:

static pair<int, int> search(const vector<int> &arr, int targetSum) {

int left = 0, right = arr.size() - 1;

while (left < right) {

int currentSum = arr[left] + arr[right];

if (currentSum == targetSum) { // found the pair

return make\_pair(left, right);

}

if (targetSum > currentSum)

left++; // we need a pair with a bigger sum

else

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

}

return make\_pair(-1, -1);

}

};

int main(int argc, char \*argv[]) {

auto result = PairWithTargetSum::search(vector<int>{1, 2, 3, 4, 6}, 6);

cout << "Pair with target sum: [" << result.first << ", " << result.second << "]" << endl;

result = PairWithTargetSum::search(vector<int>{2, 5, 9, 11}, 11);

cout << "Pair with target sum: [" << result.first << ", " << result.second << "]" << endl;

}

using namespace std;

#include <iostream>

#include <unordered\_map>

#include <vector>

class PairWithTargetSum {

public:

static pair<int, int> search(const vector<int>& arr, int targetSum) {

unordered\_map<int, int> nums; // to store number and its index

for (int i = 0; i < arr.size(); i++) {

if (nums.find(targetSum - arr[i]) != nums.end()) {

return make\_pair(nums[targetSum - arr[i]], i);

} else {

nums[arr[i]] = i; // put the number and its index in the map

}

}

return make\_pair(-1, -1); // pair not found

}

};

int main(int argc, char\* argv[]) {

auto result = PairWithTargetSum::search(vector<int>{1, 2, 3, 4, 6}, 6);

cout << "Pair with target sum: [" << result.first << ", " << result.second << "]" << endl;

result = PairWithTargetSum::search(vector<int>{2, 5, 9, 11}, 11);

cout << "Pair with target sum: [" << result.first << ", " << result.second << "]" << endl;

}

Python

def pair\_with\_targetsum(arr, target\_sum):

left, right = 0, len(arr) - 1

while(left < right):

current\_sum = arr[left] + arr[right]

if current\_sum == target\_sum:

return [left, right]

if target\_sum > current\_sum:

left += 1 # we need a pair with a bigger sum

else:

right -= 1 # we need a pair with a smaller sum

return [-1, -1]

def main():

print(pair\_with\_targetsum([1, 2, 3, 4, 6], 6))

print(pair\_with\_targetsum([2, 5, 9, 11], 11))

main()

def pair\_with\_targetsum(arr, target\_sum):

nums = {} # to store numbers and their indices

for i, num in enumerate(arr):

if target\_sum - num in nums:

return [nums[target\_sum - num], i]

else:

nums[arr[i]] = i

return [-1, -1]

def main():

print(pair\_with\_targetsum([1, 2, 3, 4, 6], 6))

print(pair\_with\_targetsum([2, 5, 9, 11], 11))

main()

Java

class PairWithTargetSum {

public static int[] search(int[] arr, int targetSum) {

int left = 0, right = arr.length - 1;

while (left < right) {

int currentSum = arr[left] + arr[right];

if (currentSum == targetSum)

return new int[] { left, right }; // found the pair

if (targetSum > currentSum)

left++; // we need a pair with a bigger sum

else

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

}

return new int[] { -1, -1 };

}

public static void main(String[] args) {

int[] result = PairWithTargetSum.search(new int[] { 1, 2, 3, 4, 6 }, 6);

System.out.println("Pair with target sum: [" + result[0] + ", " + result[1] + "]");

result = PairWithTargetSum.search(new int[] { 2, 5, 9, 11 }, 11);

System.out.println("Pair with target sum: [" + result[0] + ", " + result[1] + "]");

}

}

import java.util.HashMap;

class PairWithTargetSum {

public static int[] search(int[] arr, int targetSum) {

HashMap<Integer, Integer> nums = new HashMap<>(); // to store numbers and their indices

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

if (nums.containsKey(targetSum - arr[i]))

return new int[] { nums.get(targetSum - arr[i]), i };

else

nums.put(arr[i], i); // put the number and its index in the map

}

return new int[] { -1, -1 }; // pair not found

}

public static void main(String[] args) {

int[] result = PairWithTargetSum.search(new int[] { 1, 2, 3, 4, 6 }, 6);

System.out.println("Pair with target sum: [" + result[0] + ", " + result[1] + "]");

result = PairWithTargetSum.search(new int[] { 2, 5, 9, 11 }, 11);

System.out.println("Pair with target sum: [" + result[0] + ", " + result[1] + "]");

}

}