Two Sum

Given an array of integers, find two numbers such that they add up to a specific target number.

The function twoSum should return indices of the two numbers such that they add up to the target, where index1 must be less than index2. Please note that your returned answers (both index1 and index2) are not zero-based.

You may assume that each input would have exactly one solution.

Input: `numbers={2, 7, 11, 15}, target=9`
Output: `index1=1, index2=2`

Solution 1

```
vector<int> twoSum(vector<int> &numbers, int target)
{
    //Key is the number and value is its index in the vector.
    unordered_map<int, int> hash;
    vector<int> result;
    for (int i = 0; i < numbers.size(); i++) {
        int numberToFind = target - numbers[i];

            //if numberToFind is found in map, return them
        if (hash.find(numberToFind) != hash.end()) {
                    //+1 because indices are NOT zero based
            result.push_back(hash[numberToFind] + 1);
            result.push_back(i + 1);
            return result;
        }

            //number was not found. Put it in the map.
        hash[numbers[i]] = i;
    }
    return result;
}
```

written by naveed.zafar original link here

Solution 2

Hi, this is my accepted JAVA solution. It only go through the list once. It's shorter and easier to understand. Hope this can help someone. Please tell me if you know how to make this better :)

```
public int[] twoSum(int[] numbers, int target) {
    int[] result = new int[2];
    Map<Integer, Integer> map = new HashMap<Integer, Integer>();
    for (int i = 0; i < numbers.length; i++) {
        if (map.containsKey(target - numbers[i])) {
            result[1] = i + 1;
            result[0] = map.get(target - numbers[i]);
            return result;
        }
        map.put(numbers[i], i + 1);
    }
    return result;
}
```

written by jiaming2 original link here

Solution 3

Hello! At first glance, this can easily be solved through a quadratic algorithm BUT it can actually be done in linear time. The idea here is to use a map to keep track of the needed RIGHT operand in order for the sum to meet its target. So, we iterate through the array, and store the index of the LEFT operand as the value in the map whereas the NEEDED RIGHT operand is used as the key. When we do encounter the right operand somewhere in the array, the answer is considered to be found! We just return the indices as instructed. :]

Feel free to let me know should you have any queries for me OR if this can be improved upon!

```
public int[] twoSum(int[] nums, int target) {
        HashMap<Integer, Integer> tracker = new HashMap<Integer, Integer>();
        int len = nums.length;
        for(int i = 0; i < len; i++){
            if(tracker.containsKey(nums[i])){
                int left = tracker.get(nums[i]);
                return new int[]{left+1, i+1};
            }else{
                tracker.put(target - nums[i], i);
            }
        }
        return new int[2];
    }
```

written by waisuan original link here