

1. Two Sum

Friday, April 29, 2022 3:56 PM

LeetCode 1. Two Sum (easy)

Given an array of integers *nums* and an integer *target*, return indices of the two numbers such that they add up to *target*.

You may assume that each input would have **exactly one solution**, and you may not use the same element twice.

You can return the answer in any order.

Example 1:

Input: *nums* = [2, 7, 11, 15], *target* = 9

Output: [0, 1]

Example 2:

Input: *nums* = [3, 2, 4], *target* = 6

Output: [1, 2]

Example 3:

Input: *nums* = [3, 3], *target* = 6

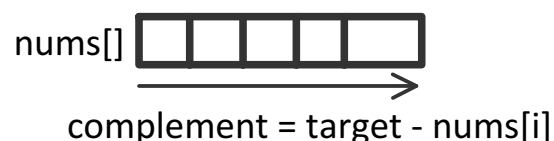
Output: [0, 1]

Solution 1 - brute force $O(n^2)$ quadratic-time

Iterate from $i = 0 \dots n-1$ through array and for each element find if the complement exists. The complement is the value needed to add to the current element in the array to equal the target value.

$$\text{complement} = \text{target} - \text{currentValue}$$

This would require a nested for loop and because we will be searching through n elements for each of the n elements in search for the complement, this would end up being a quadratic-time $O(n^2)$ solution.



Scan the *nums*[] array from left to right and for each element compute the complement and search for that complement sequentially in the *nums*[] array.

Solution 2 - double pass hashtable $O(1)$ constant-time

We can improve our solution/algorithm by improving the searching/accessing operation time from $O(n)$ linear-time to $O(1)$ constant-time with the help of a hashtable data structure.

As we see each element *and it does not already exist in the hashtable* in the *nums*[] array, we will take its index location and the element stored at that index and insert it into our hashtable.

We will then loop through the *nums*[] array and check if the complement exists in our hashtable and its relative index value.

Solution 3 - single pass hashtable $O(n)$ linear-time

We can improve our hashtable implementation by only doing a single pass through the *nums*[] array. We will add each element and its relative index position and check if the complement already exists in the hashtable... This will replicate a single-pass hashtable implementation instead of adding all values to the hashtable and then search sequentially for each element in *nums*[] if its complement exists in the hashtable.