0003\_Easy\_TwoSum\_#1\_Breakdown

Problem:

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.

Examples:

**Example 1:**

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

**Output:** [0,1]

**Explanation:** Because nums[0] + nums[1] == 9, we return [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]

Observations:

* Each input has exactly one solution.
* You may not use the same element twice.
* The answer can be returned in any order.

Questions:

* Can the array be modified?
* Is extra space allowed?

What needs to be true to solve the problem:

* Two numbers in the array need to add up to equal target.

How would a person solve the problem:

* Look at the first element in the array, look through the rest of array for a number that sums to it to equal target. Repeat with the next element and so on.

Brute force:

* Iterate through the array, for each element of the array iterate through the array looking for a number that sums to target.
* Time complexity: O(n^2), for each element of the array the array is iterated through.
  + Specifically: O(n(n-1)/2), as each array is iterated through, the previous elements are not iterated through again, resulting in smaller iterations for each element in the array.
  + Space complexity: O(1), the iterations happen in place.

Optimize (BUD, bottlenecks, unnecessary code, duplicate code):

* As a tradeoff, space can be used to create a hash map.
* Time Complexity: O(n), the array is iterated through to add to the map.
  + Specifically O(n + 1), as the array is iterated through, the map needs to be looked at for a corresponding number.
* Space Complexity: O(n): the hash map takes up at most the same amount of space as the array.

Pseudocode:

* Create a hash map to store the element of the array.
* Create an integer to look for the number that adds to the current element to equal target.
  + In other words, target minus the current element results in the element needed to be found.
* Iterate through the array to store the element in the hash map.
  + Before storing an element, check if the corresponding number exists, this avoids the current number producing a false positive.
* If no matches are found, return null.