0001\_Easy\_ContainsDuplicate\_#217\_Breakdown

Problem:

Given an integer array nums, return true if any value appears **at least twice** in the array, and return false if every element is distinct.

Questions:

* Am I allowed extra space?

What needs to be true for this problem to work:

* The array must contain no duplicate numbers.

How would a person solve this:

* Sort the array from smallest to largest, iterate through the array, return false if the number preceding is the same.
* Start at the beginning of the array, iterate through the array to look for duplicates, return false if a duplicate is found. Move through each element of the array.

Examples:

* **Example 1:**
* **Input:** nums = [1,2,3,1]
* **Output:** true
* **Example 2:**
* **Input:** nums = [1,2,3,4]
* **Output:** false
* **Example 3:**
* **Input:** nums = [1,1,1,3,3,4,3,2,4,2]
* **Output:** true

Examples takeaways:

* If an array has one or less element, every element is distinct.

Brute Force

* Start at the beginning of the array, iterate through the array to look for duplicates, return false if a duplicate is found. Move through each element of the array.
* Time Complexity: O(N^2), for each element of the array, the array is iterated through.
  + Specifically O(N((N-1)/2), the iteration of the array based on each element is decreasing, because the elements at the start of the array have already been checked.
* Space Complexity: O(1), iterating through the array happens in place.

Optimize, BUD (Bottlenecks, unnecessary work, duplicated work):

* Instead of iterating through the array, a hash table can be created to store the values of the array, this way lookup time becomes O(1).
* Instead of looping through the array to add it to the hash table and then re-looping to look for a duplicate, a duplicate can be looked for before each element of the array is added to the hash table.
* Time Complexity: O(N), worst-case the array must be fully traversed to look for a duplicate.
* Space Complexity: O(N), if the array is fully traversed, each element of the array is in the hash table.

Pseudocode:

* Initialize a hash set.
* Iterate through the array.
  + Check if the current element exists in the hash set.
    - If yes return true.
  + Add the element to the hash set.
    - (The element is added to the has set after checking to avoid a false true from the current element)
* Return false of the loop completes.

Note:

* If space is of concern, sorting the array and then iterating through it is better.
* Time Complexity: O(nlogn), algorithm used to sort the array.
  + Specifically O(nlogn + n ), the array needs to be iterated through after it is sorted.
* Space Complexity: O(1), based on the sorting algorithm.

Pseudocode for sort then compare:

* Sort the array.
* Starting at the beginning of the iterate through it, checking each proceeding digit.
  + Use ++i instead of i++, to increment I after the loop has run.
  + Return true if a match is found.
* Return false if the for loop completes.