Problem Name: Contains Duplicate

Link: https://leetcode.com/problems/contains-duplicate/

Solution Language(s): Java

Approach

In this problem we are tasked with determining if there are any duplicate numbers in the given list *nums.* In this problem, we know that we are going to have to search through the entire list, but the question comes down to which method is the fastest for doing so. In most problems where we often need to check if a set contains a value, a HashSet is a good idea. This is due to its O(1) .contains() method, thus minimizing our time complexity. My approach for this was to go through the given list and add the values to a HashSet if it does not already exist in the HashSet. If it does, return true. If we go through the entire list, we can then safely return false because there are no duplicates in the list.

Solution

```java
class Solution {
    public boolean containsDuplicate(int[] nums) {
        HashSet<Integer> dup = new HashSet<>();
        for (int i = 0; i < nums.length; i++) {
            if (dup.contains(nums[i])) {
                return true;
            } else {
                dup.add(nums[i]);
            }
        }
        return false;
    }
}
```

Time Complexity: O(n)

The time complexity in this approach is O(n), where *n* is the number of elements in *nums,* as we may have to search through the entire list if there are no duplicates.

Space Complexity: O(n)

The space complexity of this method is O(n) because we may end up adding all the elements from the array *nums* to the HashSet.

Conclusion

In conclusion, this problem is relatively simple, and contains many solutions. There is a very obvious solution which is to go through the array *nums,* and for every element check if there is a duplicate with another for loop. This results in a time complexity of O(n^2) and thus I opted to use a HashSet to keep the values of the array. This allows me to check if we have already come across the same value previously in the list with the O(1) .contains() method.