1. **Question Number and Name:**
2. [217. Contains Duplicate](https://leetcode.com/problems/contains-duplicate/)
3. **Topics:**

[Array](https://leetcode.com/tag/array/)[Hash Table](https://leetcode.com/tag/hash-table/) [Sorting](https://leetcode.com/tag/sorting/)

1. **Problem Statement:**

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.

**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

**Constraints:**

* 1 <= nums.length <= 105
* -109 <= nums[i] <= 109

1. **Code:**
2. class *Solution* {
3. public:
4. bool containsDuplicate(vector<int>*&* nums) {
5. sort(nums.begin(),nums.end());
6. bool dupli\_exist=false;
7. for(int i=0;i<nums.size()-1;i++)
8. {
9. if(nums[i]==nums[i+1])
10. {
11. dupli\_exist=true;
12. break;
13. }
14. }
15. return dupli\_exist;
16. }
17. };
    1. **Notes:**

* One solution is O(N2) solution by running two for loops over each other.
* The other solution is sorting the array first and then running a for loop iterating over the array in O(N) time complexity. This is the solution I have used above and It has O(nlogn) time complexity and O(1) space complexity since it doesnot require additional space if you discount the space taken by the sorting algorithm
* An O(N) time complexity solution would by using a hashmap and checking if an element already exists before inserting it in the map, we use the std::map.count() function to find if the key already exists, if the count is >0 it means that the key already exists in the map and there are duplicates in the array. The downside of this method is that it also require O(N) space complexity but this is the best in terms of space and time complexity as we can get.

1. **Question Number and Name:**

## [242. Valid Anagram](https://leetcode.com/problems/valid-anagram/)

1. **Topics:**

[Hash Table](https://leetcode.com/tag/hash-table/)[String](https://leetcode.com/tag/string/) [Sorting](https://leetcode.com/tag/sorting/)

1. **Problem Statement:**

Given two strings s and t, return true *if* t *is an anagram of* s*, and* false *otherwise*.

An **Anagram** is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once.

**Example 1:**

**Input:** s = "anagram", t = "nagaram"

**Output:** true

**Example 2:**

**Input:** s = "rat", t = "car"

**Output:** false

**Constraints:**

* 1 <= s.length, t.length <= 5 \* 104
* s and t consist of lowercase English letters.

1. **Code:**
2. class *Solution* {
3. public:
4. bool isAnagram(string s, string t) {
5. sort(s.begin(), s.end());
6. sort(t.begin(), t.end());
7. if(s==t)
8. return true;
9. else
10. return false;
11. }
12. };
    1. **Notes:**

* You can do this question by using hashmaps, in that case the time and space complexity will by O(S+T).
* Or you can sort the strings and check if both of them are equal or not after sorting both of them. This solution has O(nlogn) time complexity and O(1) space complexity since most interviewers assume that built-in library functions don’t consume extra memory while sorting.