Contains Duplicate

Given an array of integers, find if the array contains any duplicates. Your function should return true if any value appears at least twice in the array, and it should return false if every element is distinct.

Solution 1

This problem seems trivial, so lets try different approaches to solve it:

Starting from worst time complexity to the best one:

Time complexity: O(N^2), memory: O(1)

The naive approach would be to run a iteration for each element and see whether a duplicate value can be found: this results in O(N^2) time complexity.

```java
public boolean containsDuplicate(int[] nums) {

        for(int i = 0; i < nums.length; i++) {
            for(int j = i + 1; j < nums.length; j++) {
                if(nums[i] == nums[j]) {
                    return true;
                }
            }
        }
        return false;
    }
```

Time complexity: O(N lg N), memory: O(1) - not counting the memory used by sort

Since it is trivial task to find duplicates in sorted array, we can sort it as the first step of the algorithm and then search for consecutive duplicates.

```java
    public boolean containsDuplicate(int[] nums) {

        Arrays.sort(nums);
        for(int ind = 1; ind < nums.length; ind++) {
            if(nums[ind] == nums[ind - 1]) {
                return true;
            }
        }
        return false;
    }
```

Time complexity: O(N), memory: O(N)

Finally we can used a well known data structure hash table that will help us to identify whether an element has been previously encountered in the array.

```java
public boolean containsDuplicate(int[] nums) {

    final Set<Integer> distinct = new HashSet<Integer>();
    for(int num : nums) {
        if(distinct.contains(num)) {
            return true;
        }
        distinct.add(num);
    }
    return false;
}
```

This is trivial but quite nice example of space-time tradeoff.

written by jmnarloch original link here

Solution 2

Using anonymous set<>.
Not the most efficient as many already pointed out... but if you like one-liners ;)
akin to the solution possible with python.

```
#include <set>
using namespace std;

class Solution {
public:
    bool containsDuplicate(vector<int>& nums) {
        return nums.size() > set<int>(nums.begin(), nums.end()).size();
    }
};
```

written by chammika original link here

Solution 3

```
class Solution {
  public:
    bool containsDuplicate(vector<int>& nums) {
        return set<int>(nums.begin(), nums.end()).size() < nums.size();
    }
};
```

written by kaikai2 original link here