Contains Duplicate II

Given an array of integers and an integer *k*, find out whether there are two distinct indices *i* and *j* in the array such that **nums[i] = nums[j]** and the difference between *i* and *j* is at most *k*.

Solution 1

```
public boolean containsNearbyDuplicate(int[] nums, int k) {
        Set<Integer> set = new HashSet<Integer>();
        for(int i = 0; i < nums.length; i++){
            if(i > k) set.remove(nums[i-k-1]);
            if(!set.add(nums[i])) return true;
        }
        return false;
 }
```

written by southpenguin original link here

Solution 2

```
class Solution {
public:
    bool containsNearbyDuplicate(vector<int>& nums, int k)
    {
       unordered_set<int> s;

       if (k <= 0) return false;
       if (k >= nums.size()) k = nums.size() - 1;

       for (int i = 0; i < nums.size(); i++)
       {
           if (i > k) s.erase(nums[i - k - 1]);
           if (s.find(nums[i]) != s.end()) return true;
           s.insert(nums[i]);
       }

       return false;
    }
};
```

The basic idea is to maintain a set s which contain unique values from nums[i - k] to nums[i - 1], if nums[i] is in set s then return true else update the set.

written by luo_seu original link here

Solution 3

```
public boolean containsNearbyDuplicate(int[] nums, int k) {
    Map<Integer, Integer> map = new HashMap<Integer, Integer>();
    for (int i = 0; i < nums.length; i++) {
        if (map.containsKey(nums[i])) {
            if (i - map.get(nums[i]) <= k) return true;
        }
        map.put(nums[i], i);
    }
    return false;
}
```

written by wz366 original link here