

217. Contains Duplicate

Easy ☐ 4819 ☐ 958 ☐ Add to List ☐ Share

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 <= 10^5
- $-10^9 \le nums[i] \le 10^9$

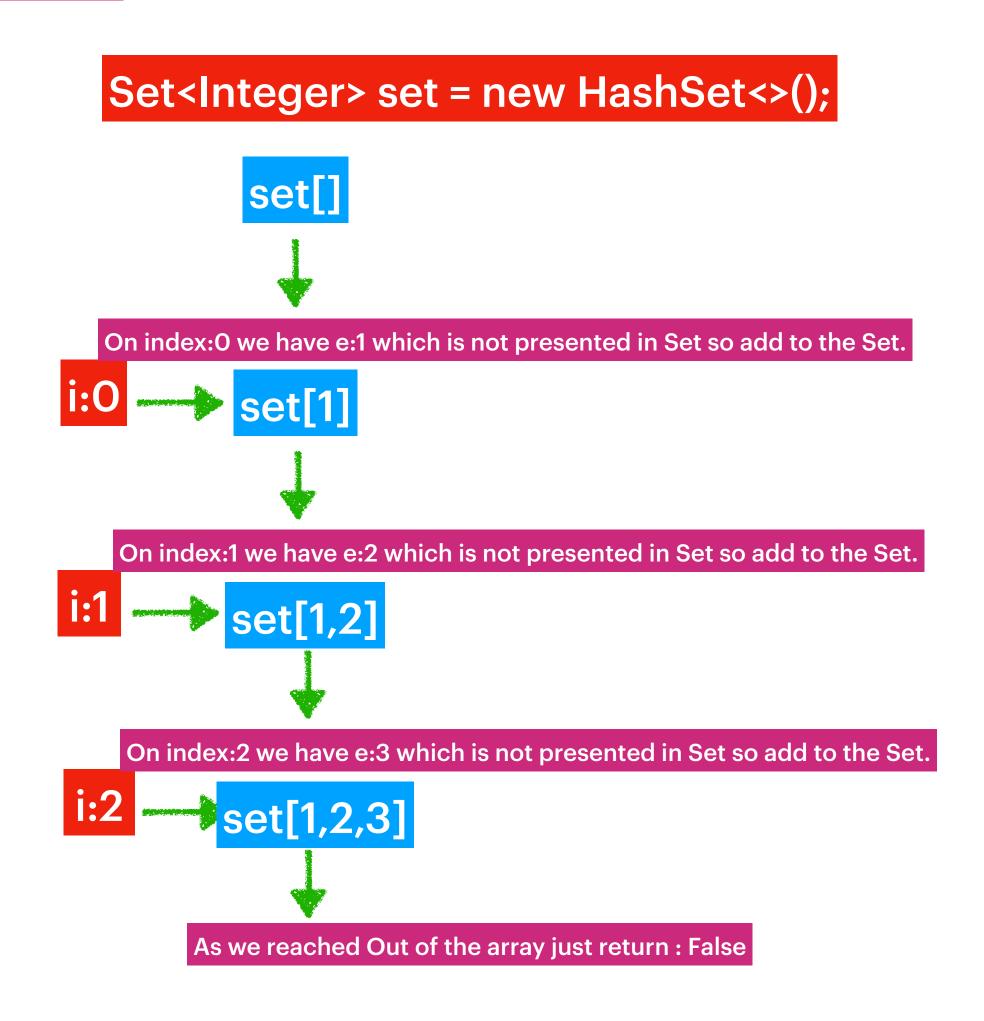
Has Duplicates should return true.

 $int[] arr = {1,2,3,1}$

Set<Integer> set = new HashSet<>(); set[] On index:0 we have e:1 which is not presented in Set so add to the Set. On index:1 we have e:2 which is not presented in Set so add to the Set. **set[1,2]** On index:2 we have e:3 which is not presented in Set so add to the Set. set[1,2,3] On index:3 we have e:1 which is presented in Set so add to the Set. Return True

Time Complexity: O(n)
Space Complexity: O(n)

Does not have Duplicates should return false. int[] arr = {1,2,3}



442. Find All Duplicates in an Array

Given an integer array nums of length n where all the integers of nums are in the range [1, n] and each integer appears **once** or **twice**, return an array of all the integers that appears **twice**.

You must write an algorithm that runs in O(n) time and uses only constant extra space.

Example 1:

```
Input: nums = [4,3,2,7,8,2,3,1]
Output: [2,3]
```

Example 2:

```
Input: nums = [1,1,2]
Output: [1]
```

Example 3:

```
Input: nums = [1]
Output: []
```

Constraints:

- n == nums.length
- $1 <= n <= 10^5$
- 1 <= nums[i] <= n
- Each element in nums appears once or twice.

```
int[] arr = {1,2,3,1,7,3,11}
```

Return all the duplicates: { 1, 3 }

Make use of List and Set Data Structures,

If the element is repeated add to List otherwise add to Set

List<Integer> list = new ArrayList<>();

Set<Integer> set = new HashSet<>();

Time Complexity: O(n)
Space Complexity: O(n)

1. Two Sum

Given an array of integers nums and an integer target, return indices of the two numbers such that they add up to target.

You may assume that each input would have *exactly* one solution, and you may not use the *same* element twice.

You can return the answer in any order.

Example 1:

```
Input: nums = [2,7,11,15], target = 9
Output: [0,1]
Explanation: Because nums[0] + nums[1] == 9, we return [0,1].
```

Example 2:

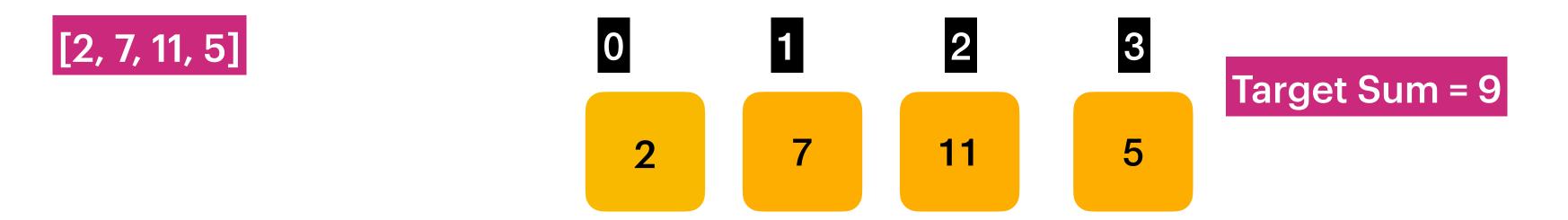
```
Input: nums = [3,2,4], target = 6
Output: [1,2]
```

Example 3:

```
Input: nums = [3,3], target = 6
Output: [0,1]
```

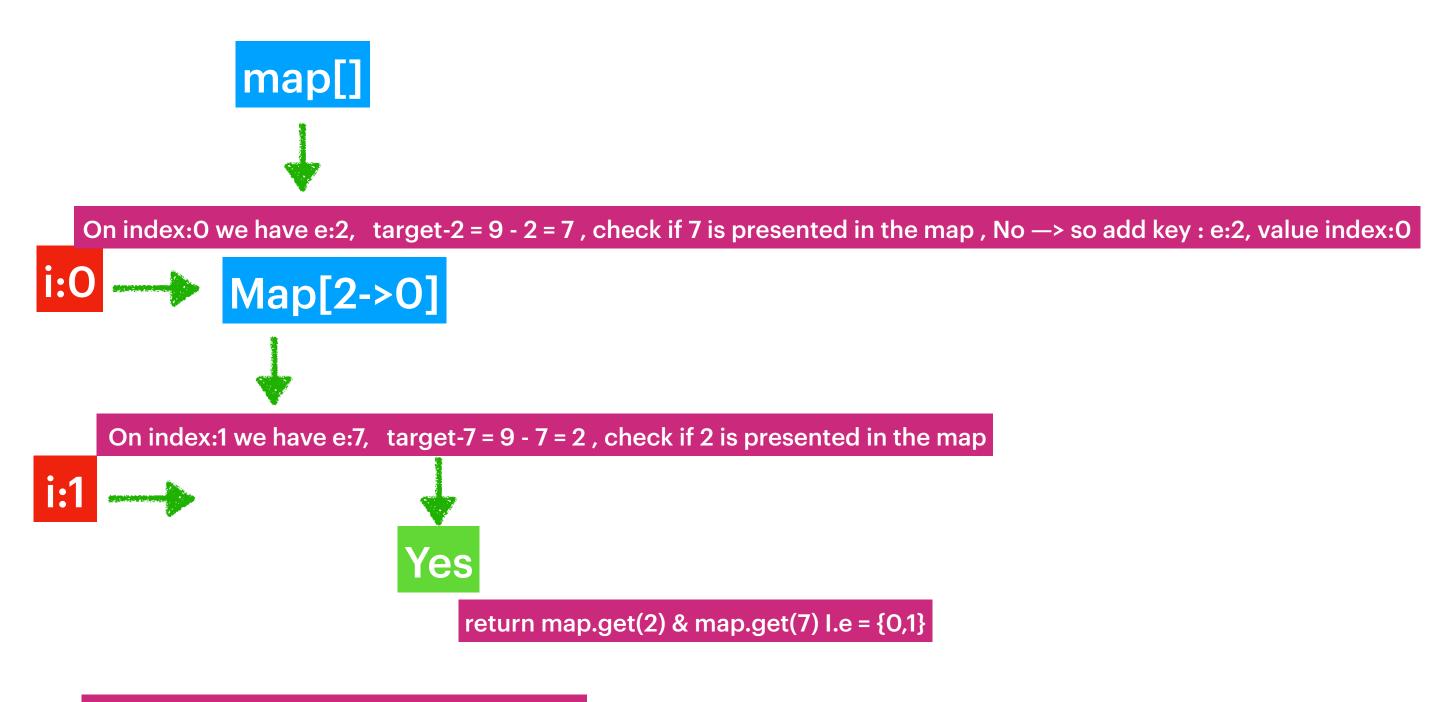
Constraints:

- 2 <= nums.length <= 10^4
- $-10^9 \le \text{nums}[i] \le 10^9$
- -10^9 <= target <= 10^9
- Only one valid answer exists.



Return the two index's summation is equals to targetSum => $nums[0] + nums[1] = 9 -> {0,1}$

Map<Integer, Integer> map = new HashMap<>();



Time Complexity: O(n)
Space Complexity: O(n)