Contains Duplicates

# Attempts	2
Date Solved	@October 21, 2025
Difficulty	Easy
Next Review	@October 28, 2025
Status	Solved
Topic/Pattern	Array and String

Link → https://neetcode.io/problems/duplicate-integer?list=neetcode150

Problem

- Check if an array has duplicates.—Return **True** if any value appears more than once, else **False**.

Example

Input	Output	Reason
[1,2,3,3]	True	3 repeats
[1,2,3,4]	False	all unique
[1,1,2,3,4,4]	True	1 and 4 repeat

Approaches

1. Brute Force — Compare every pair

```
def has_duplicate(nums):
    for i in range(len(nums)):
```

```
        for j in range(i + 1, len(nums)):
            if nums[i] == nums[j]:
                return True
    return False
```

- **Time:** O(n^2)
- **Space:** O(1) — Slow for large arrays

---

2. Sorting — Sort & check neighbors

```
def has_duplicate(nums):
    nums.sort()
    for i in range(1, len(nums)):
        if nums[i] == nums[i-1]:
            return True
    return False
```

- **Time:** O(n log n)
- **Space:** O(1) — Alters array order

---

3. Hash Set (Best) — Track seen elements

```
def has_duplicate(nums):
    seen = set()
    for n in nums:
        if n in seen:
            return True
        seen.add(n)
    return False
```

- **Time:** O(n)
- **Space:** O(n) — Fastest and cleanest

Summary

Approach	Time	Space	Notes
Brute Force	$O(n^2)$	O(1)	Simple but slow
Sorting	O(n log n)	O(1)	Needs sorting
Hash Set	O(n)	O(n)	Optimal

Edge Cases

- `[]` → False
- `[5]` → False
- `[2,2,2]` → True
- `[1,-1,2,-1]` → True

Tip

- This question tests efficiency, understanding of data structures (especially sets), and reasoning about time-space optimization.