# Question

Given an array arr of integers, check if there exists two integers N and M such that N is the double of M ( i.e. N = 2 \* M).

More formally check if there exists two indices i and j such that :

* i != j
* 0 <= i, j < arr.length
* arr[i] == 2 \* arr[j]

**Example 1:**

**Input:** arr = [10,2,5,3]

**Output:** true

**Explanation:** N = 10 is the double of M = 5,that is, 10 = 2 \* 5.

**Example 2:**

**Input:** arr = [7,1,14,11]

**Output:** true

**Explanation:** N = 14 is the double of M = 7,that is, 14 = 2 \* 7.

**Example 3:**

**Input:** arr = [3,1,7,11]

**Output:** false

**Explanation:** In this case does not exist N and M, such that N = 2 \* M.

**Constraints:**

* 2 <= arr.length <= 500
* -10^3 <= arr[i] <= 10^3

Hide Hint #1

Loop from i = 0 to arr.length, maintaining in a hashTable the array elements from [0, i - 1].

Hide Hint #2

On each step of the loop check if we have seen the element 2 \* arr[i] so far or arr[i] / 2 was seen if arr[i] % 2 == 0.

# Solution

**Q & A**:  
Q: Can you please explain this line ?

if (seen.contains(2 \* i) || i % 2 == 0 && seen.contains(i / 2))

A: i is half or 2 times of a number in seen.

**end of Q & A**

public boolean checkIfExist(int[] arr) {

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

for (int i : arr) {

if (seen.contains(2 \* i) || i % 2 == 0 && seen.contains(i / 2))

return true;

seen.add(i);

}

return false;

}

def checkIfExist(self, arr: List[int]) -> bool:

seen = set()

for i in arr:

# if 2 \* i in seen or i % 2 == 0 and i // 2 in seen:

if 2 \* i in seen or i / 2 in seen: # credit to @PeterBohai

return True

seen.add(i)

return False

**Analysis:**

Time & space: O(n), n = arr.length.