# 题目

给你一个整数数组arr，请你检查是否存在两个整数N和M，满足N是M的两倍（即N = 2 \* M）。

更正式地，检查是否存在两个下标i和j满足：

i != j

0 <= i, j < arr.length

arr[i] == 2 \* arr[j]

示例 1：

输入：arr = [10,2,5,3]

输出：true

解释：N = 10是M = 5的两倍，即10 = 2 \* 5。

示例 2：

输入：arr = [7,1,14,11]

输出：true

解释：N = 14是M = 7的两倍，即14 = 2 \* 7。

示例 3：

输入：arr = [3,1,7,11]

输出：false

解释：在该情况下不存在N和M满足N = 2 \* M。

提示：

2 <= arr.length <= 500

-10^3 <= arr[i] <= 10^3

# 分析

## 方法一：暴力法

分析：

直接遍历所有的数字对，判断一个数字是否是另一个数字的两倍。注意要用乘法判断，除法判断会有除零的问题。

代码：

class Solution {

public:

bool checkIfExist(vector<int>& arr) {

for (auto i = arr.begin(); i != arr.end(); ++i)

for (auto j = arr.begin(); j != arr.end(); ++j)

if (i != j && \*i \* 2 == \*j)

return true;

return false;

}

};

复杂度分析：

时间复杂度：O(n2)

空间复杂度：O(1)

## 方法二：排序+双指针

分析：

先对所有数字进行排序。然后维护两个指针：指针p遍历数字x，指针q寻找2x。

对于x>0 的情况，指针只需要一直前进。若q 在前进过程中找到一个比 2x大的数字，那么2x必然不存在。在 p前进的过程中 p所指向的x会不断递增，2x也会不断递增，因此指针q不需要后退。

对于x<0的情况，指针只需要一直后退。若q在后退过程中找到一个比2x小的数字，那么2x必然不存在。在p后退的过程中p所指向的x会不断递减，2x也会不断递减，因此指针q不需要前进。

代码：

class Solution {

public:

bool checkIfExist(vector<int>& arr) {

sort(arr.begin(), arr.end());

for (auto i = arr.begin(), j = arr.begin(); i != arr.end(); ++i) {

while (j != arr.end() && \*i \* 2 > \*j)

++j;

if (j != arr.end() && i != j && \*i \* 2 == \*j)

return true;

}

for (auto i = arr.rbegin(), j = arr.rbegin(); i != arr.rend(); ++i) {

while (j != arr.rend() && \*i \* 2 < \*j)

++j;

if (j != arr.rend() && i != j && \*i \* 2 == \*j)

return true;

}

return false;

}

};

**复杂度分析：**

时间复杂度： O(nlogn)

排序的时间复杂度为O(nlogn)，两次指针遍历的过程时间复杂度为O(n)，综合起来，复杂度为O(nlogn)。

空间复杂度： O(n)

Python 的sort函数空间复杂度为O(n)，双指针遍历的空间复杂度为 O(1)，综合起来，复杂度为 O(n)。

## 方法三：哈希表

分析：

先将所有数字存入哈希表，再遍历所有的数字x，判断2x 是否在哈希表中。注意数字 0 需要特殊考虑。可以通过计数来解决数字 0 的问题：若 x≠0，则 2x只需要出现一次，否则需要出现两次。

对于C++代码，由于数字范围是 [−1000，1000]，2x的范围为 [−2000，2000]我们只需要创建一个长度为4001的数组 cntcntcnt。为了解决下标为负时越界的问题，我们不直接使用数组 cntcntcnt，而是设置一个指向 cnt[2000]的指针 hash\_set作为哈希表的抽象。这样，取hash\_set[i]时会实际取用cnt[i+2000]，避免了下标越界。

说明：也可以再创建一个2x的数组，然后利用双指针查找是否在x的数组存在2x数组的元素。

代码：

class Solution {

public:

bool checkIfExist(vector<int>& arr) {

int cnt[4001] = {0};

int\* hash\_set = cnt + 2000;

for (int n : arr)

++hash\_set[n];

for (int n : arr)

if (n != 0 && hash\_set[2 \* n] >= 1)

return true;

else if (n == 0 && hash\_set[2 \* n] >= 2)

return true;

return false;

}

};

直接使用unordered\_map的方式：

class Solution {

public:

bool checkIfExist(vector<int>& arr) {

std::unordered\_map<int, int> indexMap;

// 遍历数组，将每个元素及其索引加入哈希表

for (int i = 0; i < arr.size(); ++i) {

indexMap[arr[i]] = i;

}

// 再次遍历数组，检查是否存在 arr[i] 的两倍在哈希表中

for (int i = 0; i < arr.size(); ++i) {

int doubleValue = 2 \* arr[i];

if (indexMap.find(doubleValue) != indexMap.end() && indexMap[doubleValue] != i) {

// 找到符合条件的两个下标 i 和 j

return true;

}

}

// 未找到符合条件的两个下标

return false;

}

};

**复杂度分析：**

时间复杂度：O(n)

哈希表的查询时间复杂度为O(1)，查询次数为O(n)，综合起来，时间复杂度为O(n)。

空间复杂度：O(n)

哈希表最多需要存储 n个元素。