# 题目

给你一个整数数组nums，它包含2 \* n个整数。

你需要将nums划分成n个数对，满足：

每个元素只属于一个数对。

同一数对中的元素相等。

如果可以将nums划分成n个数对，请你返回true ，否则返回false。

示例 1：

输入：nums = [3,2,3,2,2,2]

输出：true

解释：

nums中总共有6个元素，所以它们应该被划分成6 / 2 = 3个数对。

nums 可以划分成 (2, 2) ，(3, 3) 和 (2, 2) ，满足所有要求。

示例 2：

输入：nums = [1,2,3,4]

输出：false

解释：

无法将nums划分成4 / 2 = 2个数对且满足所有要求。

提示：

nums.length == 2 \* n

1 <= n <= 500

1 <= nums[i] <= 500

# 分析

## 方法一：哈希表

思路：

我们可以使用一个哈希表（例如std::unordered\_map）来统计nums中每个元素的出现次数。然后，我们遍历哈希表中的每个键值对，检查每个元素的出现次数是否为偶数。如果有任何一个元素的出现次数不是偶数，那么就无法将nums划分成满足条件的数对，返回false。如果所有元素的出现次数都是偶数，那么返回true。

首先定义了一个unordered\_map，键为nums中的元素，值为对应元素的出现次数。然后，它遍历nums数组，并使用unordered\_map统计每个元素的出现次数。接下来，它遍历unordered\_map，并检查每个值是否为偶数。如果有任何一个值不是偶数，函数返回false。如果所有值都是偶数，函数返回true。

代码：

class Solution {

public:

bool divideArray(vector<int>& nums) {

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

// 统计每个元素的出现次数

for (int num : nums) {

countMap[num]++;

}

// 检查每个元素的出现次数是否为偶数

for (const auto& pair : countMap) {

if (pair.second % 2 != 0) {

return false; // 如果有元素的出现次数不是偶数，返回false

}

}

return true; // 所有元素的出现次数都是偶数，返回true

}

};

这个解决方案的时间复杂度是O(n)，其中n是数组nums的长度，因为我们只需要遍历数组一次来填充哈希表，然后再次遍历哈希表来检查每个元素的出现次数。由于哈希表的插入和查找操作平均情况下是O(1)的，所以总的时间复杂度是线性的。