# 题目

给你一个整数数组nums，另给你一个整数original，这是需要在nums中搜索的第一个数字。

接下来，你需要按下述步骤操作：

1、如果在nums中找到original，将original乘以2，得到新original（即，令original = 2 \* original）。

2、否则，停止这一过程。

3、只要能在数组中找到新original，就对新original继续重复这一过程。

返回original的最终值。

示例 1：

输入：nums = [5,3,6,1,12], original = 3

输出：24

解释：

- 3 能在 nums 中找到。3 \* 2 = 6 。

- 6 能在 nums 中找到。6 \* 2 = 12 。

- 12 能在 nums 中找到。12 \* 2 = 24 。

- 24 不能在 nums 中找到。因此，返回 24 。

示例 2：

输入：nums = [2,7,9], original = 4

输出：4

解释：

- 4 不能在 nums 中找到。因此，返回 4 。

提示：

1 <= nums.length <= 1000

1 <= nums[i], original <= 1000

# 分析

## 方法一：排序

思路：

如果我们不对数组nums进行任何操作，那么每次更新original 后，都需要O(n)的时间完整遍历一遍。最终时间复杂度为O(n2)。我们可以对这一过程进行优化。具体而言，每次在数组中找到original后，original的数值都会比更新前更大，因此我们可以先将数组nums升序排序，这样每次更新后的original数值在数组中的位置（如有）只可能位于更新前的后面，我们只需要一边从左至右遍历排序后的nums数组一边尝试更新original即可。

代码：

class Solution {

public:

int findFinalValue(vector<int>& nums, int original) {

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

for (int num: nums) {

if (original == num) {

original \*= 2;

}

}

return original;

}

};

复杂度分析

时间复杂度：O(nlogn)，其中n为nums的长度。排序的时间复杂度为O(nlogn)，遍历更新original的时间复杂度最多为O(n)。

空间复杂度：O(logn)，即为排序的栈空间开销。

## 方法二：哈希表