# 题目

给你一个整数数组 nums ，返回数组中最大数和最小数的 最大公约数 。

两个数的 最大公约数 是能够被两个数整除的最大正整数。

示例 1：

输入：nums = [2,5,6,9,10]

输出：2

解释：

nums 中最小的数是 2

nums 中最大的数是 10

2 和 10 的最大公约数是 2

示例 2：

输入：nums = [7,5,6,8,3]

输出：1

解释：

nums 中最小的数是 3

nums 中最大的数是 8

3 和 8 的最大公约数是 1

示例 3：

输入：nums = [3,3]

输出：3

解释：

nums 中最小的数是 3

nums 中最大的数是 3

3 和 3 的最大公约数是 3

提示：

2 <= nums.length <= 1000

1 <= nums[i] <= 1000

# 分析

## 方法一：欧几里得算法

思路：

首先我们需要找到数组中的最大数和最小数。然后，我们可以使用欧几里得算法来计算这两个数的最大公约数。

欧几里得算法的基本思想是：对于两个整数a和b，它们的最大公约数等于b和a除以b的余数的最大公约数。即gcd(a, b) = gcd(b, a % b)。这个算法一直递归执行，直到余数为0，此时的除数就是最大公约数。

我们首先使用min\_element和max\_element函数找到数组中的最小值和最大值。然后，我们调用私有成员函数gcd来计算这两个数的最大公约数。gcd函数使用递归的欧几里得算法来实现。

注意：在使用这段代码之前，你需要确保包含了<algorithm>头文件，因为这个头文件中包含了min\_element和max\_element函数。同时，你还需要在类的外部或者在其他适当的地方定义gcd函数，因为它是一个私有成员函数。

代码：

class Solution {

public:

int findGCD(vector<int>& nums) {

// 找到数组中的最小值和最大值

int minNum = \*min\_element(nums.begin(), nums.end());

int maxNum = \*max\_element(nums.begin(), nums.end());

// 使用欧几里得算法计算最大公约数

return gcd(minNum, maxNum);

}

private:

int gcd(int a, int b) {

if (b == 0) {

return a;

}

return gcd(b, a % b);

}

};