# 题目

和谐数组是指一个数组里元素的最大值和最小值之间的差别正好是1。

现在，给你一个整数数组nums，请你在所有可能的子序列中找到最长的和谐子序列的长度。

数组的子序列是一个由数组派生出来的序列，它可以通过删除一些元素或不删除元素、且不改变其余元素的顺序而得到。

示例 1：

输入：nums = [1,3,2,2,5,2,3,7]

输出：5

解释：最长的和谐子序列是 [3,2,2,2,3]

示例 2：

输入：nums = [1,2,3,4]

输出：2

示例 3：

输入：nums = [1,1,1,1]

输出：0

提示：

1 <= nums.length <= 2 \* 104

-109 <= nums[i] <= 109

# 分析

## 方法一：枚举法

思路：

我们可以枚举数组中的每一个元素，对于当前枚举的元素x，它可以和x+1组成和谐子序列。我们只需要在数组中找出等于x或x+1的元素个数，就可以得到以x为最小值的和谐子序列的长度。

实际处理时，我们可以将数组按照从小到大进行排序，我们只需要依次找到相邻两个连续相同元素的子序列，检查该这两个子序列的元素的之差是否为1。

利用类似与滑动窗口的做法，begin指向第一个连续相同元素的子序列的第一个元素，end指向相邻的第二个连续相同元素的子序列的末尾元素，如果满足二者的元素之差为1，则当前的和谐子序列的长度即为两个子序列的长度之和，等于end−begin+1。

代码：

class Solution {

public:

int findLHS(vector<int>& nums) {

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

int begin = 0;

int res = 0;

for (int end = 0; end < nums.size(); end++) {

while (nums[end] - nums[begin] > 1) {

begin++;

}

if (nums[end] - nums[begin] == 1) {

res = max(res, end - begin + 1);

}

}

return res;

}

};

**复杂度分析：**

时间复杂度：O(NlogN)，其中N为数组的长度。我们首先需要对数组进行排序，花费的时间复杂度为O(NlogN)，我们需要利用双指针遍历数组花费的时间为O(2N)，总的时间复杂度T(N)=O(NlogN)+O(2N)=O(NlogN)。

空间复杂度：O(1)，需要常数个空间保存中间变量。

## 方法二：哈希表

思路：

在方法一中，我们枚举了x后，遍历数组找出所有的x和x+的出现的次数。我们也可以用一个哈希映射来存储每个数出现的次数，这样就能在O(1)的时间内得到x和x+1出现的次数。

我们首先遍历一遍数组，得到哈希映射。随后遍历哈希映射，设当前遍历到的键值对为(x,value)，那么我们就查询x+1在哈希映射中对应的统计次数，就得到了x和x+1出现的次数，和谐子序列的长度等于x和x+1出现的次数之和。

代码：

class Solution {

public:

int findLHS(vector<int>& nums) {

unordered\_map<int, int> cnt;

int res = 0;

for (int num : nums) {

cnt[num]++;

}

for (auto [key, val] : cnt) {

if (cnt.count(key + 1)) {

res = max(res, val + cnt[key + 1]);

}

}

return res;

}

};

复杂度分析：

时间复杂度：O(N)，其中N为数组的长度。

空间复杂度：O(N)，其中N为数组的长度。数组中最多有N个不同元素，因此哈希表最多存储N个数据。