# 题目

在整数数组中，如果一个整数的出现频次和它的数值大小相等，我们就称这个整数为「幸运数」。

给你一个整数数组 arr，请你从中找出并返回一个幸运数。

如果数组中存在多个幸运数，只需返回最大的那个。

如果数组中不含幸运数，则返回 -1 。

示例 1：

输入：arr = [2,2,3,4]

输出：2

解释：数组中唯一的幸运数是 2 ，因为数值 2 的出现频次也是 2 。

示例 2：

输入：arr = [1,2,2,3,3,3]

输出：3

解释：1、2 以及 3 都是幸运数，只需要返回其中最大的 3 。

示例 3：

输入：arr = [2,2,2,3,3]

输出：-1

解释：数组中不存在幸运数。

示例 4：

输入：arr = [5]

输出：-1

示例 5：

输入：arr = [7,7,7,7,7,7,7]

输出：7

提示：

1 <= arr.length <= 500

1 <= arr[i] <= 500

# 分析

## 方法一：哈希表

思路：

代码：

class Solution {

public:

int findLucky(vector<int>& arr) {

unordered\_map<int, int> mp;

for (auto &x : arr) mp[x]++;

int ans = -1;

for (auto &it : mp) {

if (it.second == it.first) {

ans = max(ans, it.first);

}

}

return ans;

}

};

复杂度分析：

记数组中的的元素个数为n，则哈希表中最多有n个键值对。

时间复杂度：遍历数组的时间代价是O(n)，遍历哈希表的时间代价也是O(n)，故渐进时间复杂度O(n)。

空间复杂度：哈希表中最多有n个键值对，故渐进空间复杂度O(n)。