《程序设计方法与艺术》课程实验报告

实验名称	1.1 查询数据						
姓　　名	傅家琪		计算机与信息学院	班　　级	计科 24-4	学　　号	2024210858
实验日期	2025.10.23		指导教师	石雷		成　　绩	

一、实验目的和要求

给定一个初始为空的可重复集容器，进行N次如下的操作(N<200000)1)插入元素x;2)删除元素x;3)查询集合中与元素x相差最小的元素.

二、解题思路和复杂度分析

数据结构选择：

选择 set 作为数据容器。set 能自动保持元素的顺序，支持插入、删除和查找操作，时间复杂度均为 O(log N)。

使用 lower_bound(x) 方法来查找集合中第一个大于等于 x 的元素，从而快速找到与 x 相差最小的元素。

三种操作的实现：

插入操作（op == 1）： 直接使用 set 的 insert() 方法将元素插入集合中。

删除操作（op == 2）： 直接使用 set 的 erase() 方法将指定元素从集合中删除。

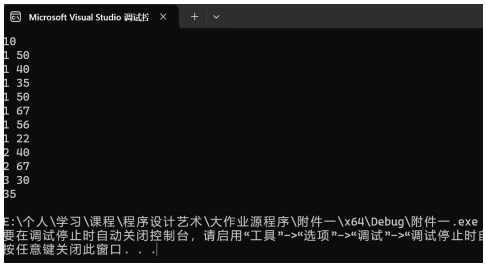
查询操作（op == 3）： 利用 lower_bound(x) 查找集合中大于等于 x 的第一个元素，再与它的前一个元素进行比较，找到差值最小的元素。需要考虑边界情况（如集合为空，或查询的元素不存在等）。

三、解题过程（流程图/伪代码）

```
// 使用 multiset 来存储元素（允许重复）
multiset<int> S
for i = 1 to N:
    read operation_type and x

    if operation_type == 1:   // 插入元素
        S.insert(x)

    else if operation_type == 2:   // 删除元素
        if S contains x:
            iterator it = S.find(x)
            S.erase(it)

    else if operation_type == 3:   // 查询与 x 相差最小的元素
        if S is empty:
            output "集合为空"
            continue
```

```
iterator it = S.lower_bound(x)
int candidate1 = (it != S.end()) ? *it : INT_MAX
int candidate2 = (it != S.begin()) ? *prev(it) : INT_MAX

int diff1 = abs(candidate1 - x)
int diff2 = abs(candidate2 - x)

if diff1 < diff2:
    result = candidate1
else if diff2 < diff1:
    result = candidate2
else:
    result = min(candidate1, candidate2)    // 差值相同时取较小的

output result
*/
```

四、实验结果及相关测试样例



《程序设计方法与艺术》课程实验报告

实验名称	1.2 约瑟夫环						
姓　　名	傅家琪		计算机与信息学院	班　　级	计科 24-4	学　　号	2024210858
实验日期	2025.10.23		指导教师	石雷		成　　绩	

一、实验目的和要求

N个小朋友们排成一个圆圈，编号分别为 1，2，3…，N;第 1 个小朋友从 1 开始报数，报到M的小朋友离开座位:然后下一个小朋友从 1 接着报数:直到剩下最后一个小朋友为止。

输入说明:输入两个数字N和M;(1<=N，M<=1000)输出说明:输出最后一个小朋友的初始编号:

二、解题思路和复杂度分析

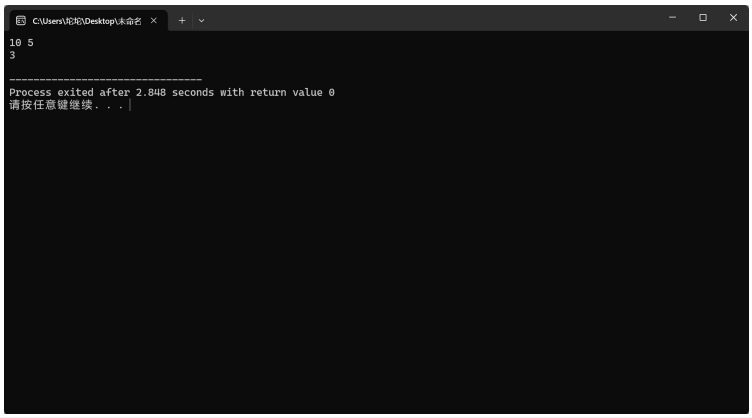
1. 使用队列模拟小朋友围成的圆圈
2. 每次报数时，将前 M-1 个小朋友从队首取出并放到队尾
3. 第 M 个小朋友直接出队（离开）
4. 重复这个过程直到队列中只剩一个小朋友

时间复杂度：O(N×M)

三、解题过程（流程图/伪代码）

```
输入 N, M

// 初始化队列
创建队列 q
for i = 1 to N:
    q.push(i)   // 将小朋友编号加入队列

// 模拟报数过程
while q.size() > 1:
    // 报数 M-1 次，将前 M-1 个小朋友移到队列末尾
    for i = 1 to M-1:
        front = q.front()
        q.pop()
        q.push(front)

    // 第 M 个小朋友离开
    q.pop()

// 输出最后剩下的小朋友
输出 q.front()
```

四、实验结果及相关测试样例



《程序设计方法与艺术》课程实验报告

实验名称	1.3 特殊质数						
姓　　名	傅家琪		计算机与信息学院	班　　级	计科 24-4	学　　号	2024210858
实验日期	2025.10.23		指导教师	石雷		成　　绩	

一、实验目的和要求

如果一个整数是质数,而且它的首位和末位也都是质数,则称之为“纯质数”。

现在，请在给出的N个正整数中统计“纯质数”的总个数。

二、解题思路和复杂度分析

遍历每个数字，先检查其是否大于 1。检查数字本身是否为质数。提取首位和末位数字，检查它们是否在质数数字集合中。统计满足所有条件的数字个数。

最坏情况时间复杂度：O(N × √max_num)

平均情况时间复杂度：通常优于最坏情况，因为许多数字可能提前被排除（如偶数或小数字）。

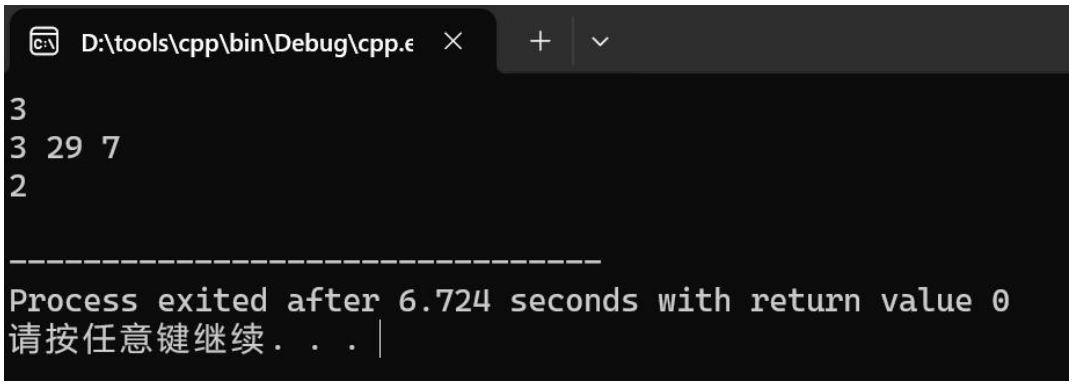
空间复杂度：O(N)用于存储输入数字，其他变量使用常数空间。

三、解题过程（流程图/伪代码）

```
开始
    // 读取输入
    输入 N
    初始化 numbers 为大小为 N 的数组

    对于 i 从 0 到 N-1:
        输入 numbers[i]

    count = 0

    // 检查每个数字
    对于 numbers 中的每个 num:
        如果 num <= 1:
            跳过当前数字，继续下一个

        // 检查数字本身是否为质数
        如果 is_prime(num) 为假:
            跳过当前数字，继续下一个

        // 提取首位和末位数字
        last_digit = num % 10
```

```
            first_digit = num

            当 first_digit >= 10:
                first_digit = first_digit / 10   // 整数除法

            // 检查首位和末位是否为质数数字
            如果 is_digit_prime(first_digit) 且 is_digit_prime(last_digit):
                count = count + 1

        // 输出结果
        输出 count
    结束
    // 质数判断函数
    函数 is_prime(n):
        如果 n <= 1:
            返回 假
        如果 n == 2:
            返回 真
        如果 n % 2 == 0:
            返回 假
        limit = 平方根(n)
        对于 i 从 3 到 limit，步长为 2:
            如果 n % i == 0:
                返回 假
        返回 真
    // 质数数字判断函数
    函数 is_digit_prime(d):
        如果 d == 2 或 d == 3 或 d == 5 或 d == 7:
            返回 真
        否则:
            返回 假
```

四、实验结果及相关测试样例



《程序设计方法与艺术》课程实验报告

实验名称	2.1						
姓　　名	傅家琪		计算机与信息学院	班　　级	计科 24-4	学　　号	2024210858
实验日期	2025.10.23		指导教师	石雷		成　　绩	

一、实验目的和要求

1:【题面描述】输出二维矩阵的四周中出现次数最多的元素，如果次数相同请按数值从大到小的次序，依次输出。

输入说明：第一行是整数 N（0<N<10000），表明矩阵的维度，接下来是 N 行，每行 N 个数。

输出说明：矩阵边界中出现次数最多的元素，如果有多个元素的出现次数相同，按数值从大到小的次序依次输出这些元素。

输入样例 1：4

1 2 3 4

2 3 4 1

3 4 1 2

4 1 2 3

输出样例 1：2

输入样例 2：4

1 2 3 4

2 3 4 1

3 4 1 2

4 1 2 1

输出样例 2：2 1

二、解题思路和复杂度分析

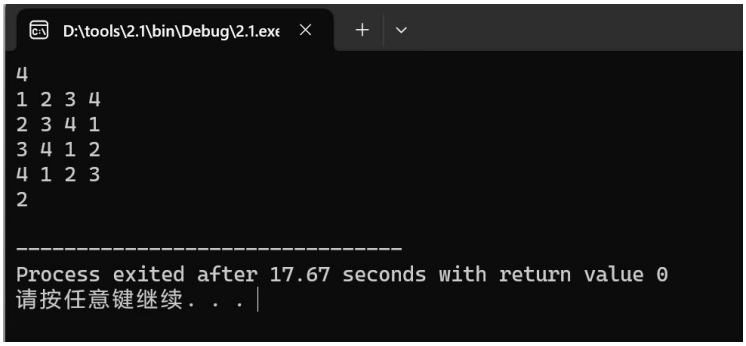
需要找出 N×N 矩阵四周边界中出现频率最高的元素，并在出现次数相同时按数值降序输出。明确边界元素的识别策略，即第一行、最后一行、第一列和最后一列的所有位置，包括四个会被重复统计的角落。在频率统计方案上，考虑到数字范围未知，选择使用映射结构来高效记录每个边界元素的出现次数。当统计完成后，找出最大频率值并收集所有达到该频率的元素，按题目要求进行降序排序，最终以连续无分隔的字符串形式输出结果。整个过程中还特别考虑了 N=1、全相同元素等边界情况.

时间复杂度：$O(N^2)$

三、解题过程（流程图/伪代码）

```
输入矩阵维度 N
初始化 N×N 矩阵 matrix
// 读取矩阵数据
对于 i 从 0 到 N-1:
    读取一行数据 line
    尝试按空格分隔解析为数字列表 row
    如果 row 长度 ≠ N:
        按字符解析 line 为数字列表 row
        调整 row 长度为 N
    matrix[i] = row
// 统计边界元素频率
初始化映射 freq
对于 i 从 0 到 N-1:
    对于 j 从 0 到 N-1:
        如果 i == 0 或 i == N-1 或 j == 0 或 j == N-1:
            freq[matrix[i][j]] += 1
// 处理特殊情况
如果 freq 为空:
    输出空字符串
    结束程序

// 找到最大频率
maxCount = 0
对于 freq 中的每个键值对 (num, count):
    如果 count > maxCount:
        maxCount = count

// 收集候选元素
初始化列表 candidates
对于 freq 中的每个键值对 (num, count):
    如果 count == maxCount:
        candidates 添加 num

// 排序候选元素
对 candidates 按降序排序

// 输出结果
对于 candidates 中的每个元素 num:
    输出 num
输出换行
```

四、实验结果及相关测试样例



《程序设计方法与艺术》课程实验报告

实验名称	2.2						
姓　　名	傅家琪		计算机与信息学院	班　　级	计科 24-4	学　　号	2024210858
实验日期	2025.10.23		指导教师	石雷		成　　绩	

一、实验目的和要求

2:【题面描述】在某国市面上流通 n 种不同面额的货币，第 i 种货币的面额为 a[i]，你可以假设每一种货币都有无穷多张。为了方便，把货币种数为 n、面额数组为 a[1..n] 的货币系统记作 (n,a)。

在一个完善的货币系统中，每一个非负整数的金额 x 都应该可以被表示出，即对每一个非负整数 x，都存在 n 个非负整数 t[i] 满足 a[i]×t[i] 的和为 x。然而， 该国度中，货币系统可能是不完善的，即可能存在金额 x 不能被该货币系统表示出。例如在货币系统 n=3，a=[2,5,9] 中，金额 1,3 就无法被表示出来。

两个货币系统 (n,a) 和 (m,b) 是等价的，当且仅当对于任意非负整数 x，它要么均可以被两个货币系统表示出，要么不能被其中任何一个表示出。现该国计划简化货币系统。他们希望找到一个货币系统 (m,b)，满足 (m,b) 与原来的货币系统 (n,a)等价，且 m 尽可能的小。他们希望你来协助完成这个艰巨的任务：找到最小的 m。

输入说明：输入文件的第一行包含一个整数 T，表示数据的组数。接下来按照如下格式分别给出 T 组数据。每组数据的第一行包含一个正整数 n。接下来一行包含 n 个由空格隔开的正整数 a[i]。

输出说明：针对输入的 T 组数据，每组输出找到的最小 m，每行输出一个。

输入样例:

```
2
4
3 19 10 6
```

二、解题思路和复杂度分析

原系统中的最小面额一定要保留（因为它是能表示的最小正金额的基础）。

如果某个面额 a[i] 可以被系统中的**比它小的面额**组合表示出来（且组合系数非负整数），那么这个 a[i] 就是冗余的，可以去掉。

最终保留下来的就是那些**不能被它前面的面额组合表示**的面额。

所以问题转化为：

1. 将 a 从小到大排序。
2. 使用完全背包的思路：设 dp[x] 表示金额 x 能否被当前已保留的面额表示。
3. 遍历排序后的 a[i]：如果 a[i] 在之前的 dp 中已经能被表示（dp[a[i]] == true），则它是冗余的，m 不

增加。否则，必须保留它，m++，并用它更新 dp 数组（完全背包过程）

排序：O(n log n)

完全背包部分：对于每个必须保留的面额 a[i]，需要 O(max(a)) 时间更新 dp 数组。

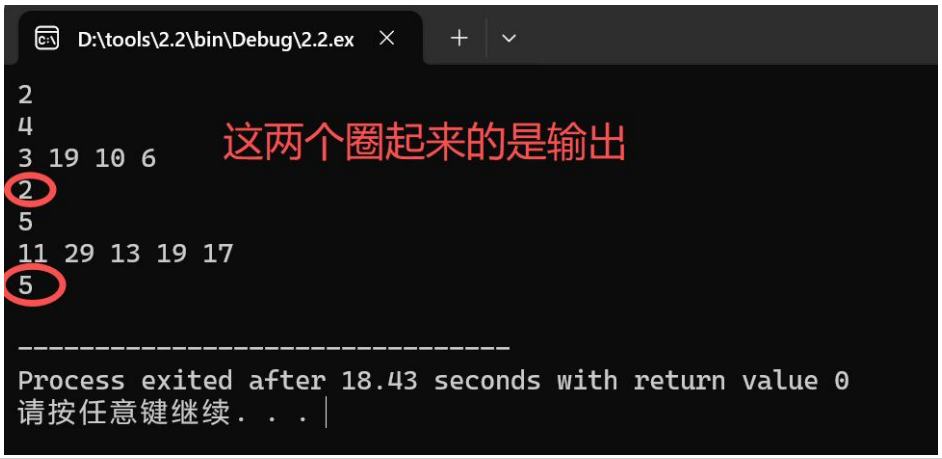
最坏情况：所有面额都保留，总复杂度 O(n * max(a))。

由于 max(a) 最大可能为 25000（题目隐含范围，常见题设），n ≤ 100，所以可接受。

三、解题过程（流程图/伪代码）

```
T = read_int()
for each group in range(T):
    n = read_int()
    a = list of n integers
    sort(a)
    max_val = a[n-1]
    dp = [False] * (max_val + 1)
    dp[0] = True
    m = 0
    for i = 0 to n-1:
        if dp[a[i]] is True:
            continue
        m += 1
        for j = a[i] to max_val:
            if dp[j - a[i]] is True:
                dp[j] = True
    print(m)
```

四、实验结果及相关测试样例



《程序设计方法与艺术》课程实验报告

实验名称	2.3						
姓　　名	傅家琪		计算机与信息学院	班　　级	计科 24-4	学　　号	2024210858
实验日期	2025.10.23		指导教师	石雷		成　　绩	

一、实验目的和要求

3 【题面描述】小明自创了一种几何绘画，每幅画的高是 N，宽是 M。画面上只有两种字符'-'和'#'。所有的'#'组成了相互不重叠的矩形。画面上的矩形相互之间不会在边上或四角相邻。请编写程序找出画面上的矩形数量。

输入说明：第一行是两个整数 N 和 M（1000 以内的正整数），表示画面的高和宽；之后的 N 行，每行 M 个字符，由字符'-'和'#'组成，表示小明画的图画。

输出说明：输出画面上的矩形数量。

输入样例 1：

```
1 10
-#-##--###
```

输出样例 1：

```
3
```

输入样例 2：

```
4  4
##-#
##--
----
#--#
```

输出样例 2：

```
4
```

二、解题思路和复杂度分析

因为矩形之间不重叠且不相邻，所以每个矩形是**一个连续的 '#' 区域，并且边界是矩形的**，不会出现 L 形等复杂形状。因此，我们只需要找到每个矩形的**左上角**即可计数。

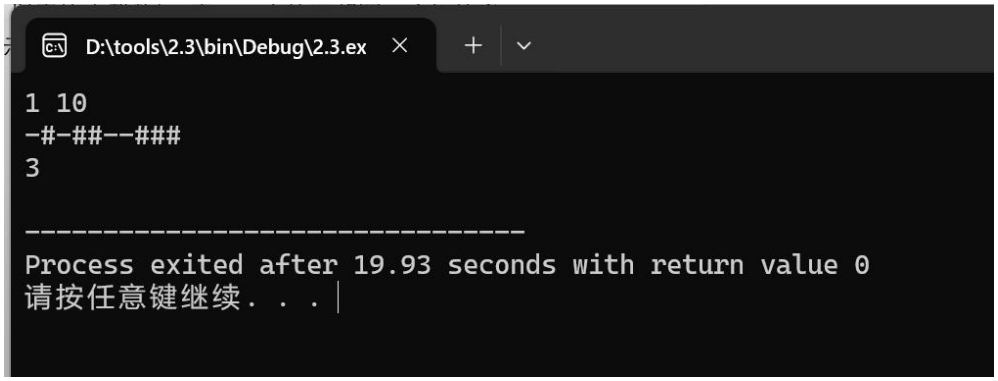
时间复杂度：O(N * M)，每个格子检查一次。

空间复杂度：O(N * M) 存储网格。

三、解题过程（流程图/伪代码）

```
 read N, M
grid = array of N strings

count = 0
for i from 0 to N-1:
    for j from 0 to M-1:
        if grid[i][j] == '#':
            if (i == 0 or grid[i-1][j] == '-') and (j == 0 or grid[i][j-1] == '-'):
                count += 1

print count
```

四、实验结果及相关测试样例



《程序设计方法与艺术》课程实验报告

实验名称	3.1						
姓　　名	傅家琪		计算机与信息学院	班　　级	计科 24-4	学　　号	2024210858
实验日期	2025.10.23		指导教师	石雷		成　　绩	

一、实验目的和要求

1【题目描述】给定一个初始为空的序列，你需要对它执行以下两种操作。
（1）从序列尾部插入元素 x
（2）查询以 i 为左/右端点，和为 x 的子段个数
（题目保证 2 操作的答案之和<=N）

输入说明：

第一行包含一个正整数 N，表示操作的次数。
接下来 N 行每行包含两个或三个整数，表示一个操作，具体如下：
（1） 1 x ：在序列尾部插入元素 x
（2） 2 i x：输出以 i 为左/右端点，和为 x 的子段个数

输出说明：

输出包含若干行，均为操作 2 输出的结果。

输入样例：

```
6
1 1
1 2
1 1
1 2
1 3
2 2 3
```

输出样例：

```
2
```

数据范围：

N<=100000；i<=已执行的操作 1 的次数；x 在 int 范围内。

二、解题思路和复杂度分析

设前缀和 prefix[k] = a[1] + ... + a[k]，prefix[0] = 0。
子段 [L, R] 的和 = prefix[R] - prefix[L-1]。
包含位置 i 的子段：$L \le i$ 且 $R \ge i$。
条件：prefix[R] - prefix[L-1] = x
即 prefix[R] - x = prefix[L-1]。
我们枚举 R 从 i 到 n，对于每个 R，需要统计有多少个 L 满足：

$L \le i$
$L-1 \le i-1$
prefix[L-1] = prefix[R] - x

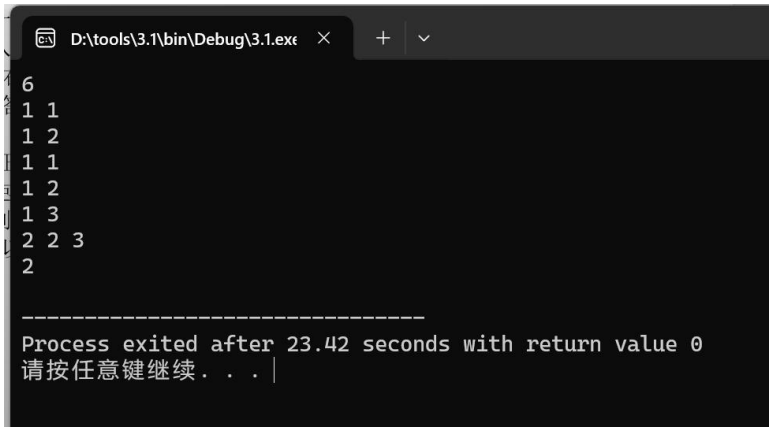
也就是：固定 R，找 L-1 在 [0, i-1] 范围内且 prefix[L-1] = prefix[R] - x 的数量。

插入：O(1) 均摊（哈希表插入）
查询：O(i log n)，但所有查询总 i 之和 ≤N，所以总 O(N log N)
空间：O(N)

三、解题过程（流程图/伪代码）

```
read N
prefix = [0]
posMap = {0: [0]}
curLen = 0

for each of N operations:
    read type
    if type == 1:
        read x
        curLen += 1
        newSum = prefix[-1] + x
        prefix.append(newSum)
        posMap[newSum].append(curLen)
    else:
        read i, x
        ans = 0
        for L from 1 to i:
            need = prefix[L-1] + x
            if need in posMap:
                arr = posMap[need]
                // 二分查找第一个 >= i 的索引
                low = 0, high = len(arr)
                while low < high:
                    mid = (low+high)/2
                    if arr[mid] < i:
                        low = mid+1
                    else:
                        high = mid
                ans += len(arr) - low
        print ans
```

四、实验结果及相关测试样例



《程序设计方法与艺术》课程实验报告

实验名称	3.2						
姓　　名	傅家琪		计算机与信息学院	班　　级	计科 24-4	学　　号	2024210858
实验日期	2025.10.23		指导教师	石雷		成　　绩	

一、实验目的和要求

2【题面描述】小明最近要负责图书馆的管理工作，需要记录下每天读者的到访情况。每位读者有一个编号，每条记录用读者的编号来表示。给出读者的来访记录，请问每一条记录中的读者是第几次出现。

输入说明：

输入的第一行包含一个整数 n，表示涛涛的记录条数。

第二行包含 n 个整数，依次表示涛涛的记录中每位读者的编号。

输出说明：

输出一行，包含 n 个整数，由空格分隔，依次表示每条记录中的读者编号是第几次出现。

输入样例

5

1 2 1 1 3

输出样例

1 1 2 3 1

二、解题思路和复杂度分析

我们需要一个**计数器**，记录每个读者编号到目前为止出现的次数。

步骤：

1. 创建一个哈希表（字典），键是读者编号，值是该编号已出现的次数。
2. 遍历输入序列：

   对于当前编号 id，如果之前没出现过，则次数为 1

   如果之前出现过，则次数 = 已出现次数 + 1

   更新哈希表

   输出当前次数

时间复杂度：O(n)：遍历一次数组，每次哈希表操作 O(1)

空间复杂度：O(n)

三、解题过程（流程图/伪代码）

```
 read n
arr = list of n integers
countMap = empty dictionary
result = empty list

for i from 0 to n-1:
    id = arr[i]
    if id not in countMap:
        countMap[id] = 1
    else:
        countMap[id] += 1
    result.append(countMap[id])

print result with spaces
```

四、实验结果及相关测试样例