习题课

TEACHING COURSEWARE POWPOINT

授课时间：2025.07.29

目录

01

题目描述

TEACHING
COURSEWARE

TEACH

题目描述

【题目描述】

Lucy家门前的大街上有 n 家店，lucy是一个喜欢收藏机械键盘的女孩子，这 n 家店中第 i 家店卖的键盘价格为 V_i ，lucy前前后后有 q 天心血来潮买电脑，其中第 i 次买电脑的预算是 K_i ，问每一次有多少店的键盘能供lucy选择，能选择当且仅当该店的键盘价格小于等于lucy的预算。

【输入格式】

输入文件第一行两个整数 n, q 。

接下来一行 n 个整数，其中第 i 个整数为 V_i 。

第 q 行每行一个整数 K 代表该次买电脑的预算。

【输出格式】

输出文件 q 行，第 i 行表示第 i 次买电脑，能有多少家店可供选择。

题目描述

【样例输入】

```
6 3
6 5 4 3 2 1
1
3
5
```

Copy

【样例输出】

```
1
3
5
```

Copy

【数据范围】

对于 30% 的数据，$n, q \leq 1000$

对于另外 30% 的数据，$n, q \leq 100000$ ，$0 \leq V_i, K_i \leq 100000$

对于 100% 的数据，$n \leq 100000$, $m \leq 100000$, $0 \leq V_i, K_i \leq 10^9$

03

题目解答

TEACHING
COURSEWARE

TEACH

题目解答

解题思路：排序预处理：先将数组按升序排序，为后续高效统计做准备。

查询排序：将所有查询按阈值 val 升序排序，使得可以按从小到大的顺序处理查询。
一次遍历统计：用一个指针 j 遍历数组，对每个查询（按排序后的顺序），移动指针统计所有 <= val 的元素个数。

由于查询已排序，当前查询的阈值 >= 上一个查询，因此指针只需向前移动，无需回溯，大幅提高效率。

恢复顺序输出：最后按查询的原始输入顺序排序，输出对应结果。

通过排序和一次线性遍历，避免了对每个查询单独遍历数组，尤其适合处理大数据量的查询场景。

离线的思路。

04

标准程序

TEACHING
COURSEWARE

TEACH

标准程序


```cpp
#include<bits/stdc++.h>
using namespace std;
#define int long long  // 将int定义为long long，避免整数溢出

int a[100005];  // 存储原始数组
// 结构体用于存储查询的相关信息
struct node {
    int val;  // 查询的数值
    int id;   // 查询的原始序号（用于最终按输入顺序输出）
    int ans;  // 该查询的答案（<=val的元素个数）
} e[100005];
// 按查询数值val升序排序的比较函数
bool cmp(node a, node b) {
    return a.val < b.val;
}
// 按查询原始序号id升序排序的比较函数（恢复输入顺序）
bool cmp2(node a, node b) {
    return a.id < b.id;
}
```

标准程序


```
signed main() {  // 因使用#define int long long，主函数用signed
    int n, q;  // n：数组元素个数；q：查询次数
    cin >> n >> q;
    // 读取数组元素
    for (int i = 1; i <= n; i++) {
        cin >> a[i];
    }
    // 对数组排序（升序），便于后续统计
    sort(a + 1, a + n + 1);
    // 读取查询信息
    for (int i = 1; i <= q; i++) {
        cin >> e[i].val;  // 查询的阈值
        e[i].id = i;      // 记录原始序号
    }
    // 按查询阈值升序排序，便于从小到大处理
    sort(e + 1, e + q + 1, cmp);
```

标准程序


```
int j = 1;  // 指向数组中当前待比较的元素（初始为第一个元素）
for (int i = 1; i <= q; i++) {
    // 继承上一个查询的答案（因查询已排序，当前阈值>=上一个，答案不会减少）
    e[i].ans = e[i - 1].ans;

    // 统计数组中所有<=当前阈值的元素
    while (j <= n && e[i].val >= a[j]) {
        j++;          // 移动指针到下一个元素
        e[i].ans++; // 答案计数+1
    }
}
// 按查询原始序号排序，恢复输入顺序
sort(e + 1, e + q + 1, cmp2);
// 输出每个查询的答案
for (int i = 1; i <= q; i++) {
    cout << e[i].ans << "\n";
}
return 0;
```