离线

TEACHING COURSEWARE POWPOINT

授课时间：2025.07.29

01

离线

TEACHING
COURSEWARE

TEACH

核心思想

先将所有查询请求一次性接收并存储，然后通过特定的排序或预处理方式，批量处理这些查询，而不是收到一个查询就立即处理一个。

02

题目思路

TEACHING
COURSEWARE

TEACH

题目思路

离线处理思路如下：
数据预处理：
先将数组a排序，为后续的高效查询做准备
离线存储查询：
将所有q个查询请求存储在结构体数组e中
每个查询不仅保存查询值val，还保存其原始序号id（用于最后按原顺序输出结果）

批量处理查询：
对所有查询按照val值进行排序（cmp函数）
对排序后的查询和排序后的数组a进行匹配：
用j指针遍历数组a
对每个查询e[i]，统计小于等于e[i].val的元素个数
利用前一个查询的结果（e[i].ans = e[i-1].ans），避免重复计算，提高效率
恢复查询顺序并输出：
按查询的原始序号id排序（cmp2函数）
按原顺序输出每个查询的结果

这种做法的优势在于：通过对查询和数据进行排序，将多次查询的处理合并为一次线性扫描大大提高了处理大量查询时的效率，这就是离线算法的典型应用场景。

03

排序浅讲

TEACHING
COURSEWARE

TEACH

题目思路

sort函数是 C++ 标准库中用于排序的工具，能快速对数组、向量等序列容器中的元素进行排序。它就像一个 “自动整理机”，可以按照我们的需求把杂乱的元素排好顺序，比如把一堆数字从小到大排列，或者把一串字符按字母顺序整理。

sort(起始地址, 结束地址的下一个位置);

如果数组arr是{5, 2, 7, 1, 3}，排序后前 3 个元素会变成{2, 5, 7}，数组整体为{2, 5, 7, 1, 3}。
比如sort(arr, arr + 4)，排序的是arr[0]、arr[1]、arr[2]、arr[3]这 4 个元素，不包含arr[4]。

起始地址：要排序的序列中第一个元素的地址。
对于数组来说，数组名就是第一个元素的地址，比如arr就表示数组arr中第一个元素的地址。

结束地址的下一个位置：这是排序范围的终点标识，指的是要排序的最后一个元素的下一个位置。对于包含n个元素的数组arr，结束地址的下一个位置就是arr + n。

例如，对数组arr的前 3 个元素排序，就可以写成：

```
sort(arr, arr + 3);
```

题目思路

sort(起始地址, 结束地址的下一个位置, cmp);

```
struct Student {
    string name;
    int age;
    int score;
};

// 排序规则：年龄小的排前面；年龄相同，分数高的排前面
bool cmp(Student a, Student b) {
    if (a.age != b.age) {
        return a.age < b.age; // 年龄升序
    } else {
        return a.score > b.score; // 分数降序
    }
}
```