上机报告-6

数算B 谢胡睿 2400014151

题目

1.题目背景

在大量学生中，查找符合某一特定条件范围的学生排名是很令人头疼的问题。

2.题目描述

对于一个成绩管理系统，初始为空，之后逐渐向其中插入学生的成绩，并查询在某一成绩范围中某一特定排名的学生成绩。

3.输入格式

第一行为一个正整数N，表示共有N个输入段。
接下来N段，每段可能为两种情况：

1. 第一行为 0，第二行为一个整数S。表示向管理系统中插入一个成绩。
2. 第一行为 1，第二行为由空格分隔的三个整数L，R和k，表示查询满足成绩在$[L,R]$范围内的第k小的成绩

4.输出格式

M行，其中M为输入中查询的数量。每行包含一个整数S，表示查询对应的成绩。

输入输出样例

输入

```
5
0
1
0
2
0
3
1
1 2 1
1
2 3 2
```

输出

```
1
3
```

数据范围和提示
对于30%的数据，$N \leq 10000$
对于50%的数据，$N \times \sqrt{N} \leq 1000000$
对于80%的数据，$Nlog_2N \leq 1000000$，且数据随机生成
对于100%的数据，$Nlog_2N \leq 1000000$，且数据可能包含极端情况

评测限时 $1s$，无存储限制
数据一定合法

Solution

总体描述

本题在线查询区间的第K小元素。由于查询涉及到当前序列&数值范围大，采用主席树+离散化的方法来解决。操作将按照输入顺序（伪在线）处理，每次插入建立一个新的树版本，查询则在最新的版本上进行。

方案：主席树与离散化

设计思路

1. **离散化**：
   - **原因**：插入的数值和查询的 L, R 范围为$long$的数据范围，直接作为线段树的下标会导致空间爆炸。需要映射。
   - **过程**：
     a. 收集所有操作中出现的数值：所有插入的 S，以及所有查询中的 L 和 R
     b. 去重与排序：将收集到的值去重并排序。在 `hw6.cpp` 中，使用了 `priority_queue` 存储负值来实现从小到大取出并去重。
     c. 建立映射：
        - `maps<long, int>`: 存储原始数值到离散化后从0开始的索引的映射。
        - `dis2val=vector<long>`: 存储离散化索引到原始数值的反向映射。
     - `data_num` 将表示不同值的总数，线段树将在 `[0, data_num-1]` 这个离散化后的索引范围上构建。
2. **可持久化线段树 (主席树)**：
   - **版本管理**
   - **节点结构 (`Node`)**
   - **更新操作 (`update`)**
   - **区间和查询 (`sum`)**
   - **查询第k小 (`kth_smallest`)**
3. **处理操作与查询逻辑 (伪在线)**：
   - 将所有插入操作存入 `nums` (pair<数值, 原始操作顺序>)，查询操作存入 `querys`。
   - 遍历原始操作顺序（从0到 $N_{ops}-1$）。通过比较 `nums` 中下一个待处理插入和 `querys` 中下一个待处理查询的原始操作顺序，来决定当前应该执行插入还是查询。
   - **执行插入 `0 S`**：
     a. `num_insert++` ，版本号增加。
     b. 调用 `update(root[num_insert-1], root[num_insert], 0, data_num-1, maps[S])` 来构建新版本的树。
   - **执行查询 `1 L R K`**：
     a. 获取当前最新的树根：`last_root = root[num_insert]`。
     b. 获取查询参数 L_{orig} 的离散化索引：`l_discrete = maps[L_{orig}]`。
     c. 计算在当前版本中，值严格小于 L_{orig} 的元素数量：`count_less_L = 0; if(l_discrete > 0) countlessL = sum(last_root, 0, data_num-1, 0, l_discrete-1);`。
     d. 调整 K：`k_adjusted = K + count_less_L`。这个 `k_adjusted` 代表了要找的数在所有数中的全局排名
     e. 使用主席树查询全局第 `k_adjusted` 小的数的离散化索引：`ans_discrete = kth_smallest(last_root, 0, data_num-1, k_adjusted)`。
     f. 通过反向映射 `dis2val` 得到原始数值并输出：`printf("%ld\n", dis2val[ans_discrete]);`。

优缺点

优点：

高效处理历史版本查询：**不想动脑**想别的。
对数级复杂度：单次插入和查询的时间复杂度均为 $O(\log M_{discrete})$，其中 $M_{discrete}$ 是离散化后不同值的数量。总时间复杂度约为

$O(N_{ops} \log M_{discrete})$。
空间可接受：每次更新只增加 $\log M_{discrete}$ 个新节点，总空间复杂度为 $O(N_{ops} \log M_{discrete})$。

缺点：

1. exhausting
2. 感觉离散化开销还是很大，但是不离散化包不行（还是有什么奇怪的地方我没注意到）

问题与挑战

离散化

伪在线处理

总结

通过实现主席树+离散化，解决动态插入序列中查询区间第K小元素的问题。