成绩单

使用哈希表记录姓名到堆中下标的映射。

堆的实现参考课本最大堆，有删除、插入、维护操作。在课本堆的基础上添加了每次交换元素时更新哈希表中数据的操作，该操作复杂度为 $O(1)$ 。

输出分数根据姓名进行 $O(1)$ 找到学生在堆中的下标，直接输出对应的分数。

输出最高分的所有人则是递归地判断堆顶及其儿子的成绩是否与堆顶相同，如果相同继续递归，如果不同，不递归。

保证额外的访问次数不超过 k ，总访问次数不超过 $2k$ 。

输出时根据字典序排序，可以考虑使用基数排序 $O(k)$ 。

红黑树

红黑树主要部分完全按照课本实现，区别之处在于以下三个操作：

- 查询第 k 大
- 查询后继
- 查询个数

这三个操作均可通过记录每个结点对应的子树大小来实现 $O(\log n)$ 的算法。

每个结点维护一个 `siz` 属性，记录对应的子树大小。

每当添加或删除结点时，从修改的结点自下而上更新路径上所有结点的 `siz` 值。

查询第 k 大和查询后继的算法与课本二叉搜索树一章给出的算法完全相同。

查询个数的算法可以先求出两边界的后继和前驱，求后继和前驱结点的排名，两排名相减得出结果。

前驱和后继求法相近，不再赘述。

求排名算法类似于查询第 k 大，根据二叉搜索树的性质递归查询。

最长递增子序列

`LowerBound` 使用二分查找返回数组中第一个不小于给定值的元素的下标。

`dp[i]` 表示长度为 `i + 1` 的递增子序列末尾元素的最小值。

一个事实是：对于任意长度为 `i + 1` 的递增子序列，一定存在一个长度为 `i` 的递增子序列，且后者的末尾元素小于前者的末尾元素。

因此 `dp` 数组可以保证始终有序，严格证明已在第四次作业中给出。

对于 `a` 数组中的每一个数，都可以将其视为某个递增子序列的末尾元素，使用 `LowerBound` 在 `dp` 数组中查找以其作为末尾元素的最长递增子序列的倒数第二个元素的大小以及序列长度。

时间复杂度为 $O(T(\mathrm{LowerBound}(n)) \times n) = O(n \log n)$ 。