双指针经典题

新·电子对撞

解题思路

- 递增/严格递增：二分?
- 我最开始的思路是针对每个 B_i 找它是否在 $A_1 \ldots A_n$ 也出现过，再进行之后的合并。但是这需要新开 2 个标记数组，分别标记每个 A_i、B_i 是否合并过、如果合并了合并后的系数是多少，最后还需要再整合两个数组的数据，虽然时间复杂度最多不会超过 $O(n \log n + m \log m)$，但是写起来很复杂
- 实际上最后的“整合两个数组的数据”的方法就是双指针，那莫不如上来就双指针

解题思路

如何双指针

- 初始化对于 A 的指针 $p_1 = 0$，对于 B 的指针 $p_2 = 0$
- 当 p_1, p_2 均未指向队列末尾时
  - 如果 $A[p_1] < B[p_2]$，应向合并后表达式末尾追加 $A[p_1]$ 次项，系数为 $a[p_1]$，并令 p_1 后移一位
  - 如果 $A[p_1] > B[p_2]$，应向合并后表达式末尾追加 $B[p_2]$ 次项，系数为 $b[p_2]$，并令 p_2 后移一位
  - 如果 $A[p_1] = B[p_2]$ 且合并后系数不为 0，应向合并后表达式末尾追加 $A[p_1]$(或$B[p_2]$) 次项，系数为 $a[p_1] + b[p_2]$，并令 p_1, p_2 同时后移一位
- 最后，如果 p_1 没指到 A 末尾，把 A 中剩余的所有项追加到表达式末尾；否则把 B 中剩余的所有项追加到表达式末尾

解题思路

复杂度

- 时间复杂度：$O(n+m)$，a, A, b, B 序列的每一项都只被访问一次
- 空间复杂度：$O(n+m)$，来源于输入的 a, A, b, B 序列和存储结果的 c, C 序列

其他解题思路

- 为什么不让用C++和Java
- 因为这道题最简单的做法是直接用C++里的map、Java里的HashMap，时间复杂度最多只会乘一个对数级别的数（由容器带来），在 100000 量级的数据条件下一般也能过