领奖台数

使用归并排序作为基础，添加了两个数组记录顺序对个数和逆序对个数（实际上还需要额外的 `temp` 数组用于存放这两者在排序过程中计算出的中间结果，而这些临时数组与算法本身关系不大，下面不再说明）。

`orderHead[i]` 记录已合并的区间中，以 `a[i]` 为头的顺序对的数量。

`reverseDf[i]` 记录已合并的区间中，以 `a[i]` 为头的逆序对数量减去以 `a[i]` 为尾的逆序对数量。

则对于每次合并，分两种情况讨论。

如果待合并元素位于左区间，视为 $i < j < mid < k$，领奖台数需要增加

$$\#(\text{以}a_i\text{为头的顺序对数量}) \times \#(\text{右侧比}a_i\text{小的数的个数})$$

上式第一个因子为 `orderHead[i]` 的值。

如果待合并元素位于右区间，视为 $i < mid < j < k$，领奖台数需要增加

$$\#(\text{以}a_j\text{为头的逆序对数量}) \times \#(\text{左侧比}a_j\text{小的数的个数}) - \#(\text{以}a_k\text{为尾的逆序对数量}) \times \#(\text{左侧比}a_k\text{小的数的个数})$$

由于 j 和 k 不相关，上式可简化为

$$(\#(\text{以}a_j\text{为头的逆序对数量}) - \#(\text{以}a_j\text{为尾的逆序对数量})) \times \#(\text{左侧比}a_j\text{小的数的个数})$$

上式第一个因子为 `reverseDf[j]` 的值。

逆序对与顺序对的求法在此前作业中已给出，不再赘述。

该算法在归并排序的基础上并未增添其他循环或递归，因此时间复杂度为 $O(n\log n)$ 。

最短达标区间

计算数组的前缀和（直接使用数组原地址），由于数组中有负数，其前缀和并非递增。

假设最短达标区间为 $[l, r]$,

如果 l 位于前缀和的递减区间，则可以将 l 向后移动，直至 l 位于一个极值点处，此时区间长度显然更短，且区间和更大。

对 r 同理。

因此得证： l 和 r 一定位于前缀和的递增区间内。

使用 `vec` 作为单调栈，记录递增的前缀和，并对每一个元素 `a[r]` 求满足 `sum[l] + sum[r] >= k` 的最大 `l` 。

对所有 `r` ，求最小的 `r - l + 1` ，即为最短区间长度。

找零

使用 `ans[j]` 表示找 `j` 元钱的方案数。

并且由于外层循环是对货币面额 `coins[i]` 的枚举，循环过程中 `ans[j]` 实际上意味着对于前 `i` 种货币，找 `j` 元钱的方案数。

该方案数等于对于前 `i - 1` 种货币的方案数加上增加了第 `i` 种货币的方案数，即 `ans[j] = ans[j] + ans[j - coins[i]]` 。

两重循环时间复杂度为 $O(Vn)$ 。

最大连续子方阵

`line[i][j]` 表示第 `i` 行中以 `a[i][j]` 为终点的最长连续序列。

`row[i][j]` 表示第 `j` 列中以 `a[i][j]` 为终点的最长连续序列。

`ans[i][j]` 表示以 `a[i][j]` 为终点的最大连续方阵。

则显然当 `a[i][j] = 1` 时，行列值均加一，方阵大小视连续行列值和对角线上最大连续方阵大小而定。

两重循环，时间复杂度 $O(n^2)$ 。