

四、附加题（根据题目要求写出解答过程，共 10 分）。

为了在算法基础课程期中考试取得好成绩，小花同学在考前狂刷《算法导论》课后习题。在不刷任何题的情况下，小花能够考 S 分。《算法导论》中共有 n 道题，刷第 i 题能使她的期中考试成绩提升 s_i ($s_i \geq 0$)。但与此同时，刷题会产生疲劳，进而影响考试发挥，如果小花一共刷了 k 道题，则她会因为疲劳而损失 αk 分 ($\alpha \geq 0$)。小花不喜欢跳着刷题，也不喜欢倒着刷题，更不喜欢重复刷题，因此她只会从某一题开始刷连续的若干题。假设小花希望让自己的最终分数落入 $[l, r]$ ，请问她有多少种刷题方案？试设计一个具有 $O(n \lg n)$ 最坏时间复杂度的算法解决该问题。（在小花所处的平行世界中，考试的分数未必是 $0 \sim 100$ 的整数，因此你不能使用计数排序等对数据范围有限制的方法）

解：令 $D_k = \sum_{i=1}^k s_i - \alpha k$ ，当小花刷题的范围是 $(x, y]$ 时，她的最终得分为 $S + D_y - D_x$ 。因此问题可转化为：满足 $x < y$ 且 $D_y - D_x \in [l - S, r - S]$ 的数对 $\langle x, y \rangle$ 的个数。与逆序对数量类似，这可以在归并排序的过程中顺便统计，或者通过维护一棵红黑树进行统计。详见[区间和的个数](#)

四、附加题（根据题目要求写出解答过程，共 10 分）。

小迪作为一名投资者，常常研究不同的投资策略。最近，他提出了“价值区间”分析法：给定某只股票在一定时间范围内的价格 $a = [a_0, a_1, \dots, a_{n-1}]$ ($a_i > 0$)，区间大小 W ，以及价值差 A ，若存在 $i < j \in [0, n-1]$ ，且 $|i - j| \leq W$ ， $|a_i - a_j| \leq A$ ，则称该股票存在“价值区间” (i, j) 。

例如， $a = [1, 3, 5, 10]$ ，若 $W = 2$ ， $A = 2$ ，则存在两个“价值区间”： $(0, 1)$ ， $(1, 2)$ ；若 $W = 2$ ， $A = 1$ ，则不存在“价值区间”。

请你为小迪设计一个时间复杂度为 $O(n)$ 的算法 `VALUERANGE`，以判断给定的股票是否存在“价值区间”。给出算法的伪代码，并分析其时间复杂度。（输入： a ， W ， A ；输出：True or False）

Algorithm 7 `VALUERANGE( $a, W, A$ )`

```

1:  $map = \{\}$ 
2:  $size = A + 1$ 
3: for  $i = 0$  to  $n - 1$  do
4:    $idx = \lfloor \frac{a_i}{size} \rfloor$ 
5:   if  $idx$  in  $map$  then
6:     return True
7:    $l = idx - 1$ 
8:    $r = idx + 1$ 
9:   if  $l$  in  $map$  and  $|a_i - map[l]| \leq A$  then
10:    return True
11:   if  $r$  in  $map$  and  $|a_i - map[r]| \leq A$  then
12:    return True
13:    $map[idx] = a_i$ 
14:   if  $i \geq k$  then
15:      $pop\ map[\lfloor \frac{a_{i-k}}{size} \rfloor]$ 
16: return False

```

时间复杂度 $O(n)$ ，空间复杂度 $O(k)$ 。

四、附加题（根据题目要求写出解答过程，共10分）。

你得到了 n 块按顺序排成一排的「记忆碎片」，第 i 块碎片上记录的记忆发生时间为 t_i 。已知 $t_1, t_2, \dots, t_n$ 是 $1 \sim n$ 的一个排列，也就是说，这 n 块碎片本来应该是按 $1, 2, \dots, n$ 的时间顺序出现的，但现在被打乱了。你希望通过若干次交换操作，把这些碎片恢复成严格单调递增的顺序：

$$(t_1, t_2, \dots, t_n) \longrightarrow (1, 2, \dots, n).$$

你拥有一个「回忆能力值」 k 。一次操作中，你可以任选两块碎片，设它们记忆的时间分别是 x 和 y ，当且仅当 $|x - y| \leq k$ 时，你可以交换它们现在所在的位置。交换的体力消耗规则如下：

- 若 $|x - y| = 1$ ，也就是这两段记忆发生在相邻的两个时间点上，由于发生时间过于靠近的两块碎片很难回想，这次交换需要消耗你 **1 点** 精力；
- 若 $2 \leq |x - y| \leq k$ ，则这次交换**不需要**消耗精力，代价为 **0**。

你可以进行任意多次满足条件的交换操作。请你计算：为了将碎片恢复成 $(1, 2, \dots, n)$ ，你最少需要消耗多少点精力。

设计时间复杂度尽可能小的算法。（输入： $n, [t_1, t_2, \dots, t_n], k$ ；输出：最少需要消耗的精力值）

解：下面分情况讨论。

一、 $n < 4$ 的情况 排列的状态总数不超过 $3! = 6$ 个。直接写出 6 种排列的最优代价表。不再赘述。

二、 $n \geq 4$ 的情况

(1) $k = 1$ 这是经典的“只用相邻数值换位”把一个排列排成升序，所需的最少此类交换次数等于这个排列的逆序对个数。形式化地，定义

$$\text{inv}(t) = \#\{(i, j) \mid 1 \leq i < j \leq n, t_i > t_j\}.$$

那么当 $k = 1$ 时，答案就是

$$\boxed{\text{ans} = \text{inv}(t)}.$$

可以用归并排序维护 $\text{inv}(t)$ 计数器，可以在 $O(n \log n)$ 时间内统计。下面是算法：

1. 若当前这段只剩下一个元素，则它本身已经有序，逆序对数为 0。
2. 否则，把当前这段成长度尽量相等的左右两半，递归地分别对左半段、右半段做同样的处理，这一步会得到“左半段内部的逆序对数”和“右半段内部的逆序对数”，同时也会得到“排好序的左半段”和“排好序的右半段”。
3. 接下来要把两个已经排好序的子段“像归并排序那样”合成一段有序的数组。合并时，左半段和右半段都从头开始往后看：

- 如果左半段当前的数 $\leq$ 右半段当前的数，就先把左半段的这个数放进结果里，然后左半段指针后移；
 - 否则（也就是左半段当前的数 $>$ 右半段当前的数），说明右半段这个数应该排在当前左半段这个数以及左半段后面所有还没放进去的数的前面。由于左半段目前还没放进去的元素都是“比这个右半段元素大”的，所以这一时刻就一口气产生了“左半段当前还剩多少个元素”这么多个逆序对；把这个数量加到总答案里，再把右半段当前的数放进结果，然后右半段指针后移。
4. 两边有一边先用完了，就把另一边剩下的所有元素按顺序接到结果后面；这一部分不会再产生新的逆序对。
5. 当前这段的逆序对数 = “左半段内部的逆序对数” + “右半段内部的逆序对数” + “合并时数到的那些跨左右的逆序对数”。

(2) $k = 2$ 当 $|x - y| = 2$ 时交换免费；而差为 2 的两个数一定同奇偶，于是：

- 所有奇数记忆 $(1, 3, 5, \dots)$ 之间可以免费任意重排；
- 所有偶数记忆 $(2, 4, 6, \dots)$ 之间也可以免费任意重排；
- 想要把一个“奇数位置上却放了偶数”的错误修正，就必须和某个“偶数位置上却放了奇数”的错误做一次 $|x - y| = 1$ 的付费交换。

目标序列 $(1, 2, \dots, n)$ 在位置 i 上的奇偶为 $i \bmod 2$ 。统计原序列中奇偶不匹配的位置数：

$$\text{mism} = \#\{i \mid (t_i \bmod 2) \neq (i \bmod 2)\}.$$

一条付费交换（交换一对差为 1 的数，例如 5 和 6）会同时把一个“奇放偶位”的错误和一个“偶放奇位”的错误一起消掉，因此所需的付费次数正好是

$$\boxed{\text{ans} = \frac{\text{mism}}{2}}.$$

在 $n \geq 4$ 的前提下，奇偶不匹配的个数一定是偶数，上式必为整数。该步骤只需线性时间 $O(n)$ 。

(3) $k \geq 3$ 当能力值至少为 3 时，我们既可以免费交换差为 2 的一对（保持奇偶），也可以免费交换差为 3 的一对（改变奇偶）；当 $n \geq 4$ 时，可以证明任意一个数字都能通过若干次免费交换移动到任意位置，因此不需要做任何一次差为 1 的付费交换，答案为

$$\boxed{\text{ans} = 0}.$$