seg

原题

加强了一下，缩短了时限并且要求强制在线。

预处理出所有交点，然后对于每个线段，发现耗能值只有 $O(n)$ 次改变。

直接计算出这些改变的值，然后记录对应位置挂在线段树上处理询问即可。注意询问的 l, r 需要一次二分找到它在线段树上的位置。

时间复杂度 $O(n^2 + q \log n)$。可能略微卡常。

data

原题

首先我们抛开我们需要求最大值这个问题，我们先考虑假如说被打乱的数据 S' 是定的，我们如何计算最小交换次数。

考虑我们对于每一个位置设定一个 a_i，表示这个位置的字符最终要去到哪个地方。不难发现一次操作最多把 a_i 序列的逆序对数减一，并且除非 a 序列满足 $a_i = i$，一定存在一种操作使得逆序对数减少。所以我们可以知道最小的交换次数等价于 a_i 的逆序对数量。

然后我们再考虑如何对于一个 $S' \to S$ 的过程设定 a_i。容易发现对于同色的位置 i, j，我们保证 $a_i < a_j$ 一定最优，因为交换任何一个 i, j 显然逆序对数只增不减。

最后我们再考虑什么样的 S' 是最大的。考虑同色位置 $i, j (i < j)$ 的 a_i, a_j 中间关于逆序对的一些性质。为了方便我们设 (i, j) 区间内没有同色的位置。考虑区间中间的 a_k 的贡献。

- 若 $a_k > a_j$，那么逆序对贡献为 1。
- 若 $a_k < a_j$，那么逆序对贡献也为 1。
- 其他情况时逆序对贡献为 0。

注意到其他情况时，如果我们把 i, j 挪到 $i, i+1$ 或者 $j-1, j$ 都会有贡献，考虑挪在一起的时候前两个情况的贡献。发现如果挪到 $i, i+1$，那么情况一的贡献变成 0，情况二的贡献变成 2；挪到 $j-1, j$ 则反之。不难发现我们有以下挪法保证不劣：

- 若 $a_k > a_j$ 的 k 值比 $a_k < a_j$ 的 k 值多，那么挪到 $j-1, j$。
- 若 $a_k > a_j$ 的 k 值比 $a_k < a_j$ 的 k 值少，那么挪到 $i, i+1$。
- 若相等，那么挪到哪里都不会劣。

挪完之后，我们就把这两个挨在一起的字母看做是一个，再去模拟上面我们挪的过程，最后序列就一定会变成一个相同字母都是挨在一起的序列。

所以根据上面的推导，我们得到了一个重要的结论：存在一个 S' 满足其最小交换次数最大，并且所有相同的字母挨在一起。

那么这道题就可以直接枚举字母的排列，然后树状数组计算逆序对。时间复杂度 $O(|\Sigma|! \sum n \log n)$。可以通过。

本题有预处理 $O(|\Sigma| n)$，然后 $O(|\Sigma|^2)$ 计算逆序对的方式，可以参考洛谷的题解区。

plane

原题

首先观察到答案是 $O(n)$ 级别的，因为一定有一个子序列的峰在整个序列的最大值的位置。

然后考虑我们一次只计算最大值右侧的贡献，最大值左侧的贡献显然可以由翻转序列再做一次得到。

我们先来尝试判定峰点对是否合法。设峰点对为 (x,y)，不难发现 $[1,x]$ 和 $[y,n]$ 的部分是好做的。以 $[1,x]$ 的部分为例，我们可以设 f_i 表示对于 $[1,i]$ 中的元素，如果第一个子序列最后的位置选了 i，第二个子序列最后位置的最小值。转移如下：

- $a_i > a_{i-1}$，$f_i \leftarrow f_{i-1}$。
- $a_i > f_{i-1}$，$f_i \leftarrow a_{i-1}$。

我们设 $[y,n]$ 部分的转移使用 g，含义就是对于 $[i,n]$ 中的元素，如果第一个子序列最开始的位置是 i，第二个子序列最开始位置的最小值。

然后考虑中间的部分，这部分问题抽象下来就是这样：问序列是否可以被分成一个上升的子序列和一个下降的子序列。其实这还是一个两个子序列组合的问题，我们可以仿照我们对 $[1,x]$ 的处理设计状态。

设 $h_{i,1}$ 表示对于 $[x,i]$ 的元素，上升的序列选择了 i，下降序列末尾的最大值；$h_{i,2}$ 表示对于 $[x,i]$ 的元素，下降的序列选择了 i，上升序列末尾的最小值。其实容易发现和上面几乎完全一致，上面只有一个状态只是因为两个拼合在一起的子序列性质相同可以互换。

h 的转移如下：

- $a_i > a_{i-1}$，$h_{i,1} \leftarrow h_{i-1,1}$。
- $a_i < a_{i-1}$，$h_{i,2} \leftarrow h_{i-1,2}$。
- $a_i > h_{i-1,2}$，$h_{i,1} \leftarrow a_{i-1}$。
- $a_i < h_{i-1,1}$，$h_{i,2} \leftarrow a_{i-1}$。

最后只需要判定 $h_{y,1} > g_y$ 即可。容易发现处理出 f,g,h 之后所有判定都是 $O(1)$ 的，所以总时间复杂度线性。

monkey

原题

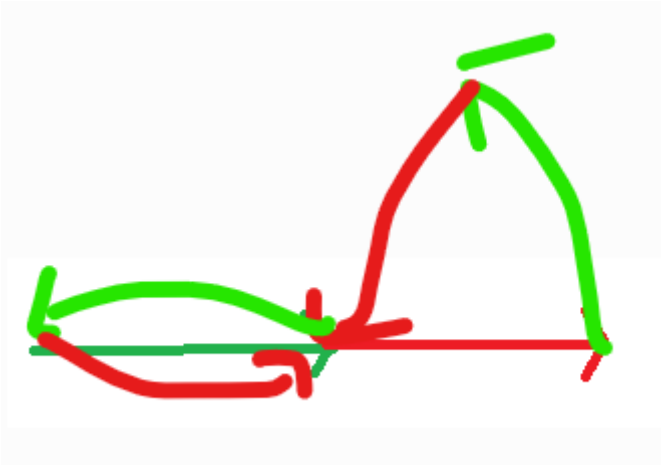
一句题外话：这个题原来想出 $n,m \le 2 \times 10^5$，最后对一些性质研究了很久，然后就没时间造 down 和数据了。现在看来这个题如果 $n,m \le 3000$ 放在 OI 赛制里面是不是就成简单题了啊/ng。

首先发现如果仅存在模式一那就是欧拉回路板子题。所以首先看这个图能不能直接用欧拉回路做。

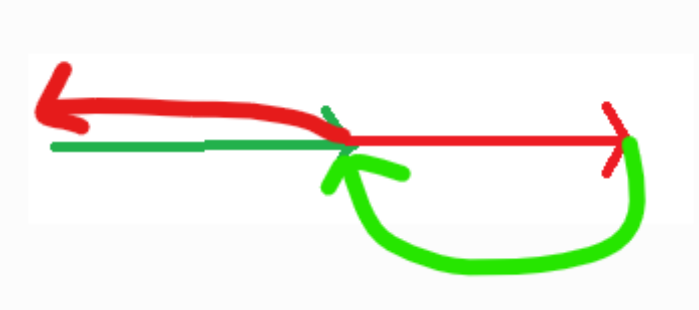
考虑在模式二情况下有哪些图可以被破坏。画一个图稍微理解一下：

图中绿色的边表示走过但是没有被破坏的边，以下简称绿边；红色的边表示被破坏的边，以下简称红边。

假设我们模式二就像上图一样，即把一条非红色的边变绿之后再把另外一条边变红。为了把所有边变红，我们一定会回到左边那条绿边的其中一个端点然后把这个边变红。考察这个回到的过程，不难发现这里一定会产生绿边：



这是一种极限情况，最后我们回到了中间的点，然后又回到了最开始的那个模型。



这是另外一个极限情况，最后我们到达了最左边的点，然后又回到了最开始的模型。

不难发现其他情况都是弱于这两种情况，也一定会出现最开始的模型。这里陷入了一个循环，说明这个图根本删不完。

所以我们证明了，如果把一条边变绿，那么我们必须立马回头把刚刚我们走的边变红，否则一定会遗留一些边删不掉。这等价于模式二只能删菊花。

那么这个题就比较好做了。考虑贪心，枚举菊花的中心，枚举这个中心的所有出边，如果这个出边对应的点是奇点，那么直接把这条出边删掉，然后判断该图是否连通（除度数为 0 的非菊花中心的点）。删完一圈之后，如果该图具有欧拉回路，并且最后可以使得欧拉路终点在菊花中心处，那么可以直接构造一个欧拉回路，然后在菊花中心处切换模式直接破坏剩余的一个菊花。时间复杂度 $O((n+m)^2)$。