# 题目选讲：动态规划 & 数论

清华大学　任舍予

2024 年 8 月 9 日

**问题 (CF1647F)**

给定排列 $[p_1, \ldots, p_n]$，将其恰好划分为两个单峰子序列，求两个单峰子序列的峰的组合的情况数。

数据范围：$2 \leq n \leq 5 \times 10^5$。

- 容易发现最大值一定是其中一个子序列的峰。记最大值的位置为 $p$，不妨设另一个峰的位置为 $q > p$。

- 容易发现最大值一定是其中一个子序列的峰。记最大值的位置为 $p$，不妨设另一个峰的位置为 $q > p$。
- 对于 $[1, p-1]$ 中的数，应让不在以 $p$ 为峰的子序列中的最大值尽可能小，$[q+1, n]$ 一部分类似。

- 容易发现最大值一定是其中一个子序列的峰。记最大值的位置为 $p$，不妨设另一个峰的位置为 $q > p$。
- 对于 $[1, p-1]$ 中的数，应让不在以 $p$ 为峰的子序列中的最大值尽可能小，$[q+1, n]$ 一部分类似。
- 对于 $[p+1, q-1]$ 中的数，需要划分成两个有值域限制的递增和递减序列。

- 容易发现最大值一定是其中一个子序列的峰。记最大值的位置为 $p$，不妨设另一个峰的位置为 $q > p$。
- 对于 $[1, p-1]$ 中的数，应让不在以 $p$ 为峰的子序列中的最大值尽可能小，$[q+1, n]$ 一部分类似。
- 对于 $[p+1, q-1]$ 中的数，需要划分成两个有值域限制的递增和递减序列。
- 以上三部分都可以简单线性 DP 解决。

- 容易发现最大值一定是其中一个子序列的峰。记最大值的位置为 $p$，不妨设另一个峰的位置为 $q > p$。
- 对于 $[1, p-1]$ 中的数，应让不在以 $p$ 为峰的子序列中的最大值尽可能小，$[q+1, n]$ 一部分类似。
- 对于 $[p+1, q-1]$ 中的数，需要划分成两个有值域限制的递增和递减序列。
- 以上三部分都可以简单线性 DP 解决。
- 时间复杂度 $O(n)$，空间复杂度 $O(n)$。

## 问题 (CF908E)

给定 $m$，令 $M = 2^m - 1$。给定 $\{0, 1, \ldots, M\}$ 的大小为 $n$ 的子集 $T$，定义集合 $T \subseteq S \subseteq \{0, 1, \ldots, M\}$ 是好的当且仅当：

- $a \in S \implies a \operatorname{xor} M \in S$；
- $a, b \in S \implies a \operatorname{and} b \in S$。

求好的集合的个数。

数据范围：$1 \le m \le 1000$，$1 \le n \le \min(2^m, 50)$。

- 由于取反与按位与足够构造出或与异或，$S$ 中的元素一定构成一个线性空间，因此只需要对位进行集合划分。

- 由于取反与按位与足够构造出或与异或，$S$ 中的元素一定构成一个线性空间，因此只需要对位进行集合划分。
- 给定 $T$ 的限制相当于限制了某些位不能在同一个集合中，可以将所有出现情况相同的位的方案数直接相乘。

- 由于取反与按位与足够构造出或与异或，$S$ 中的元素一定构成一个线性空间，因此只需要对位进行集合划分。
- 给定 $T$ 的限制相当于限制了某些位不能在同一个集合中，可以将所有出现情况相同的位的方案数直接相乘。
- 对于位的集合划分，只需要简单 DP，枚举最后一个数所在的集合大小然后乘组合数转移即可。

- 由于取反与按位与足够构造出或与异或，$S$ 中的元素一定构成一个线性空间，因此只需要对位进行集合划分。
- 给定 $T$ 的限制相当于限制了某些位不能在同一个集合中，可以将所有出现情况相同的位的方案数直接相乘。
- 对于位的集合划分，只需要简单 DP，枚举最后一个数所在的集合大小然后乘组合数转移即可。
- 时间复杂度 $O(m^2+mn)$，空间复杂度 $O(m^2+mn)$。

**问题 (CF1209E2)**

给定 $n \times m$ 的矩阵，可以对每一列进行若干次循环移位，求操作完成后每一行的最大值之和的最大值。

数据范围：$1 \le n \le 12$，$1 \le m \le 2000$。

- 注意到一定可以取到每一列的最大值的前 $n$ 大之和，于是只有 $n$ 列是有用的。

- 注意到一定可以取到每一列的最大值的前 $n$ 大之和，于是只有 $n$ 列是有用的。
- 考虑状压最大值已经计算的行，转移时枚举当前列哪些位置对哪些行产生贡献即可。

- 注意到一定可以取到每一列的最大值的前 $n$ 大之和，于是只有 $n$ 列是有用的。
- 考虑状压最大值已经计算的行，转移时枚举当前列哪些位置对哪些行产生贡献即可。
- 时间复杂度 $O(mn + n^2 2^n + n3^n)$，空间复杂度 $O(n2^n)$。

## 问题 (CF908G)

定义 $S(n)$ 为将 $n$ 所有数位从小到大排序后得到的数，求 $\sum_{i=1}^{n} S(i)$。
数据范围：$1 \le n \le 10^{700}$。

- 没有办法直接数位 DP 的原因在于插入一个数字的贡献不可计算。

- 没有办法直接数位 DP 的原因在于插入一个数字的贡献不可计算。
- 由于每种数字的出现是连续的一段，且插入一个数只会让整体平移，于是考虑对每种数字分别计算贡献。

- 没有办法直接数位 DP 的原因在于插入一个数字的贡献不可计算。
- 由于每种数字的出现是连续的一段，且插入一个数只会让整体平移，于是考虑对每种数字分别计算贡献。
- DP 时维护当前若干位的数字排完序的贡献，转移时有三类情况：

- 没有办法直接数位 DP 的原因在于插入一个数字的贡献不可计算。
- 由于每种数字的出现是连续的一段，且插入一个数只会让整体平移，于是考虑对每种数字分别计算贡献。
- DP 时维护当前若干位的数字排完序的贡献，转移时有三类情况：
  - 选择更小的数字：在高位插入，没有贡献。

- 没有办法直接数位 DP 的原因在于插入一个数字的贡献不可计算。
- 由于每种数字的出现是连续的一段，且插入一个数只会让整体平移，于是考虑对每种数字分别计算贡献。
- DP 时维护当前若干位的数字排完序的贡献，转移时有三类情况：
  - 选择更小的数字：在高位插入，没有贡献。
  - 选择更大的数字：在低位插入，贡献为 $\times 10$。

- 没有办法直接数位 DP 的原因在于插入一个数字的贡献不可计算。
- 由于每种数字的出现是连续的一段，且插入一个数只会让整体平移，于是考虑对每种数字分别计算贡献。
- DP 时维护当前若干位的数字排完序的贡献，转移时有三类情况：
  - 选择更小的数字：在高位插入，没有贡献。
  - 选择更大的数字：在低位插入，贡献为 $\times 10$。
  - 选择当前数字：在中间位插入，贡献不确定。

- 没有办法直接数位 DP 的原因在于插入一个数字的贡献不可计算。
- 由于每种数字的出现是连续的一段，且插入一个数只会让整体平移，于是考虑对每种数字分别计算贡献。
- DP 时维护当前若干位的数字排完序的贡献，转移时有三类情况：
  - 选择更小的数字：在高位插入，没有贡献。
  - 选择更大的数字：在低位插入，贡献为 $\times 10$。
  - 选择当前数字：在中间位插入，贡献不确定。
- 于是还需要另一个 DP 计算插入一个当前数字的贡献和，转移只需要考虑上面两类。

- 没有办法直接数位 DP 的原因在于插入一个数字的贡献不可计算。
- 由于每种数字的出现是连续的一段，且插入一个数只会让整体平移，于是考虑对每种数字分别计算贡献。
- DP 时维护当前若干位的数字排完序的贡献，转移时有三类情况：
  - 选择更小的数字：在高位插入，没有贡献。
  - 选择更大的数字：在低位插入，贡献为 $\times 10$。
  - 选择当前数字：在中间位插入，贡献不确定。
- 于是还需要另一个 DP 计算插入一个当前数字的贡献和，转移只需要考虑上面两类。
- 时间复杂度 $O(\log_{10} n)$，空间复杂度 $O(\log_{10} n)$。

**问题 (洛谷 P8321)**

给定序列 $[A_1, \ldots, A_n], [B_1, \ldots, B_n]$，求

$$\sum_{p \in S_n} \prod_{i=1}^{n} \min(A_i, B_{p_i}).$$

数据范围：$1 \leq n \leq 5000$。

- 从排列角度无法做任何计算，于是只能从权值角度每种数计算贡献。

- 从排列角度无法做任何计算，于是只能从权值角度每种数计算贡献。
- 考虑按值域设计 DP，计算前若干小的数产生的贡献以及对之后的数的影响。

- 从排列角度无法做任何计算，于是只能从权值角度每种数计算贡献。
- 考虑按值域设计 DP，计算前若干小的数产生的贡献以及对之后的数的影响。
- 每加入一个数时，只需要讨论其与之前的数匹配还是与之后的数匹配。

- 从排列角度无法做任何计算，于是只能从权值角度每种数计算贡献。
- 考虑按值域设计 DP，计算前若干小的数产生的贡献以及对之后的数的影响。
- 每加入一个数时，只需要讨论其与之前的数匹配还是与之后的数匹配。
- 可以发现，状态只需要记录任意一个匹配情况即可推出其他的匹配情况。

- 从排列角度无法做任何计算，于是只能从权值角度每种数计算贡献。
- 考虑按值域设计 DP，计算前若干小的数产生的贡献以及对之后的数的影响。
- 每加入一个数时，只需要讨论其与之前的数匹配还是与之后的数匹配。
- 可以发现，状态只需要记录任意一个匹配情况即可推出其他的匹配情况。
- 时间复杂度 $O(n^2)$，空间复杂度 $O(n^2)$。

## 问题 (CF1415F)

数轴上有一个人，每个时刻可以让自己的位置 $\pm 1$ 或不动。
当他位于整点上时，他可以放置一个无法移动的分身，同时摧毁已经存在的分身（如果存在），放置分身不消耗时间。
有 $n$ 个任务，第 $i$ 个任务要求他或者他的分身 $t_i$ 时刻在 $x_i$ 处。判断是否存在完成所有任务的方案。
数据范围：$1 \le n \le 5000$。

- 考虑直接 DP，钦定第 $i$ 个时刻由本人完成，并记录当前分身所在的位置。

- 考虑直接 DP，钦定第 $i$ 个时刻由本人完成，并记录当前分身所在的位置。
- 仅有一种情况无法转移与统计答案，即先完成 $1 \sim i-2$ 的所有任务，然后到 $i-1$ 放置克隆，同时本人到 $i$ 完成任务。

- 考虑直接 DP，钦定第 $i$ 个时刻由本人完成，并记录当前分身所在的位置。
- 仅有一种情况无法转移与统计答案，即先完成 $1 \sim i-2$ 的所有任务，然后到 $i-1$ 放置克隆，同时本人到 $i$ 完成任务。
- 只需要再增加一个 DP 计算完成 $1 \sim i-1$ 所有任务后到 $i$ 放置克隆的最小时间即可。

- 考虑直接 DP，钦定第 $i$ 个时刻由本人完成，并记录当前分身所在的位置。
- 仅有一种情况无法转移与统计答案，即先完成 $1 \sim i-2$ 的所有任务，然后到 $i-1$ 放置克隆，同时本人到 $i$ 完成任务。
- 只需要再增加一个 DP 计算完成 $1 \sim i-1$ 所有任务后到 $i$ 放置克隆的最小时间即可。
- 时间复杂度 $O(n^2)$，空间复杂度 $O(n^2)$。

### 问题

给定长度为 $n$ 的小写字母串 $S$，求有多少个长度为 $n$ 的小写字母串 $T$ 满足 $\mathrm{LCS}(S,T) \geq n-k$。

数据范围：$2 \leq n \leq 5 \times 10^4$，$0 \leq k \leq 3$。

- 考虑计算 LCS 的 DP：

$$f_{i,j} = \max(f_{i-1,j}, f_{i,j-1}, f_{i-1,j-1} + [T_i = S_j]).$$

- 考虑计算 LCS 的 DP：

$$f_{i,j} = \max(f_{i-1,j}, f_{i,j-1}, f_{i-1,j-1} + [T_i = S_j]).$$

- 可以设计如下 DP：记 $g_{i,w_0,\dots,w_n}$ 表示 $T[1\dots i]$ 对应的 DP 值分别为 $w_0,\dots,w_n$ 的方案数，转移只需要枚举 $T_{i+1}$。

- 考虑计算 LCS 的 DP：

$$f_{i,j} = \max(f_{i-1,j}, f_{i,j-1}, f_{i-1,j-1} + [T_i = S_j]).$$

- 可以设计如下 DP：记 $g_{i,w_0,\ldots,w_n}$ 表示 $T[1\ldots i]$ 对应的 DP 值分别为 $w_0,\ldots,w_n$ 的方案数，转移只需要枚举 $T_{i+1}$。
- 注意到 $f_{i,j} - f_{i,j-1} \in \{0,1\}$，因此只有 $|j-i| \le k$ 的 DP 状态是有用的，且 $f_{i,i} \ge i-k$ 才会贡献答案。
- 因此所有有效的 $g$ 的下标只有 $O(k4^k)$ 种，直接转移可以做到 $O(nk^3 4^k)$。
- 由于转移只与 DP 值和新加入字符可以匹配的位置相关，因此可以预处理每种状态匹配每种位置的转移结果。
- 时间复杂度 $O(k16^k + nk^2 4^k)$，空间复杂度 $O(16^k + nk4^k)$。

**问题 (联合省选 2022 最大权独立集问题)**

给定一棵 $n$ 个点的二叉树，每个点有点权 $a_i$。每次操作时选择一条边 $(u, v)$，交换 $a_u, a_v$，然后删去 $(u, v)$，代价为 $a_u + a_v$。求总代价的最小值。
数据范围：$2 \le n \le 5000$。

- 朴素的 DP 为：设 $f_{u,x,y}$ 表示子树 $u$ 内 $a_x$ 换出去，$a_y$ 换进来的最小代价。

- 朴素的 DP 为：设 $f_{u,x,y}$ 表示子树 $u$ 内 $a_x$ 换出去，$a_y$ 换进来的最小代价。
- $y$ 一维的状态空间过大，于是需要优化状态，改为记录换进来的点移动到的位置。

- 朴素的 DP 为：设 $f_{u,x,y}$ 表示子树 $u$ 内 $a_x$ 换出去，$a_y$ 换进来的最小代价。
- $y$ 一维的状态空间过大，于是需要优化状态，改为记录换进来的点移动到的位置。
- 根据 $(u,l_u),(u,r_u),(u,f_u)$ 三条边断开的顺序，有六种不同的转移方式。

- 朴素的 DP 为：设 $f_{u,x,y}$ 表示子树 $u$ 内 $a_x$ 换出去，$a_y$ 换进来的最小代价。
- $y$ 一维的状态空间过大，于是需要优化状态，改为记录换进来的点移动到的位置。
- 根据 $(u, l_u), (u, r_u), (u, f_u)$ 三条边断开的顺序，有六种不同的转移方式。
- 简单整理后可以发现，每类转移均可以通过前缀最小值优化。优化后枚举的复杂度与树形背包一致。

- 朴素的 DP 为：设 $f_{u,x,y}$ 表示子树 $u$ 内 $a_x$ 换出去，$a_y$ 换进来的最小代价。
- $y$ 一维的状态空间过大，于是需要优化状态，改为记录换进来的点移动到的位置。
- 根据 $(u, l_u), (u, r_u), (u, f_u)$ 三条边断开的顺序，有六种不同的转移方式。
- 简单整理后可以发现，每类转移均可以通过前缀最小值优化。优化后枚举的复杂度与树形背包一致。
- 时间复杂度 $O(n^2)$，空间复杂度 $O(n^2)$。

## 问题 (LOJ 4081)

定义 $\mathrm{mex}(S)$ 表示集合 $S$ 中未出现的最小自然数。

定义 $\mathrm{ultra}(S) = \{a \mid a \operatorname{xor} m \in S\}$，其中 $m = \mathrm{mex}(S) - 1$，xor 表示按位异或。

设 $A_0 \subseteq \{0, 1, 2, \ldots, 2^k - 1\}$ 且 $0 \in A_0$。对于 $i \geq 1$，令 $A_i = \mathrm{ultra}(A_{i-1})$。

若存在自然数 $l$，使得对于任意 $i \geq l$，都有 $\mathrm{mex}(A_i) = \mathrm{mex}(A_l)$，则称 $\mathrm{mex}(A_l)$ 为集合 $A_0$ 的极限。

$T$ 组询问，给定 $k, n, p$，求满足以下要求的集合 $A_0$ 的个数：

1. $A_0 \subseteq \{0, 1, 2, \ldots, 2^k - 1\}$ 且 $0 \in A_0$；
2. $|A_0| = n$；
3. $A_0$ 的极限为 $p$。

数据范围：$1 \leq T \leq 10^5$，$1 \leq k \leq 17$，$1 \leq n < 2^k$，$1 \leq p \leq 2^k$。

- 考虑将集合 $S$ 中的所有元素插入 Trie 中进行分析，设当前处理的最高位为 $k$:

- 考虑将集合 $S$ 中的所有元素插入 Trie 中进行分析，设当前处理的最高位为 $k$：
- 若左子树不为满二叉树，则 $\mathrm{mex}(S) - 1 < 2^k$，这样一次 ultra 操作并不会改变元素所处的子树，即右子树的分布与 $S$ 的极限无关。

- 考虑将集合 $S$ 中的所有元素插入 Trie 中进行分析，设当前处理的最高位为 $k$：
- 若左子树不为满二叉树，则 $\mathrm{mex}(S)-1<2^k$，这样一次 ultra 操作并不会改变元素所处的子树，即右子树的分布与 $S$ 的极限无关。
- 若左子树为满二叉树，且 $2^k\in S$，则 $\mathrm{mex}(S)-1\geq 2^k$，这样一次 ultra 操作会交换左右子树，可以递归至右子树中解决。

- 考虑将集合 $S$ 中的所有元素插入 Trie 中进行分析，设当前处理的最高位为 $k$：
- 若左子树不为满二叉树，则 $\mathrm{mex}(S) - 1 < 2^k$，这样一次 ultra 操作并不会改变元素所处的子树，即右子树的分布与 $S$ 的极限无关。
- 若左子树为满二叉树，且 $2^k \in S$，则 $\mathrm{mex}(S) - 1 \geq 2^k$，这样一次 ultra 操作会交换左右子树，可以递归至右子树中解决。
- 若左子树为满二叉树，且 $2^k \notin S$，$2^{k+1} - 1 \in S$，则 $\mathrm{mex}(S) - 1 = 2^k - 1$，这样一次 ultra 操作相当于分别翻转左右子树，使 $\mathrm{mex}(S)$ 变大，可以递归至右子树中解决。

- 考虑将集合 $S$ 中的所有元素插入 Trie 中进行分析，设当前处理的最高位为 $k$：
- 若左子树不为满二叉树，则 $\mathrm{mex}(S)-1<2^k$，这样一次 ultra 操作并不会改变元素所处的子树，即右子树的分布与 $S$ 的极限无关。
- 若左子树为满二叉树，且 $2^k\in S$，则 $\mathrm{mex}(S)-1\geq 2^k$，这样一次 ultra 操作会交换左右子树，可以递归至右子树中解决。
- 若左子树为满二叉树，且 $2^k\notin S$，$2^{k+1}-1\in S$，则 $\mathrm{mex}(S)-1=2^k-1$，这样一次 ultra 操作相当于分别翻转左右子树，使 $\mathrm{mex}(S)$ 变大，可以递归至右子树中解决。
- 若左子树为满二叉树，且 $2^k\notin S$，$2^{k+1}-1\notin S$，则 $\mathrm{mex}(S)$ 恒为 $2^k$，即集合 $S$ 的极限为 $2^k$。

- 根据上述计算极限的过程，不难发现集合的极限必定为 $2$ 的次幂。

- 根据上述计算极限的过程，不难发现集合的极限必定为 $2$ 的次幂。
- 固定极限 $p = 2^q$，考虑根据递归的过程进行计数 DP：

- 根据上述计算极限的过程，不难发现集合的极限必定为 $2$ 的次幂。
- 固定极限 $p = 2^q$，考虑根据递归的过程进行计数 DP：
- 设 $f_{i,n}$ 表示大小为 $n$，极限为 $p$ 的 $\{0, 1, \ldots, 2^i - 1\}$ 的子集的数量；

- 根据上述计算极限的过程，不难发现集合的极限必定为 $2$ 的次幂。
- 固定极限 $p = 2^q$，考虑根据递归的过程进行计数 DP：
- 设 $f_{i,n}$ 表示大小为 $n$，极限为 $p$ 的 $\{0, 1, \ldots, 2^i - 1\}$ 的子集的数量；
- 设 $g_{i,n}$ 表示大小为 $n$，极限为 $p$ 的 $\{0, 1, \ldots, 2^i - 2\}$ 的子集的数量；

- 根据上述计算极限的过程，不难发现集合的极限必定为 $2$ 的次幂。
- 固定极限 $p = 2^q$，考虑根据递归的过程进行计数 DP：
- 设 $f_{i,n}$ 表示大小为 $n$，极限为 $p$ 的 $\{0, 1, \ldots, 2^i - 1\}$ 的子集的数量；
- 设 $g_{i,n}$ 表示大小为 $n$，极限为 $p$ 的 $\{0, 1, \ldots, 2^i - 2\}$ 的子集的数量；
- 初始化：$f_{q+1,2^q+n} = g_{q+1,2^q+n} = \binom{2^q-2}{n}$

- 转移:

$$\begin{cases} f_{i,n} \times \binom{2^i}{r} \to f_{i+1,n+r} \\ f_{i,n} \times \binom{2^i-1}{r} \to g_{i+1,n+r} \\ f_{i,n} \to f_{i+1,n+2^i} \\ g_{i,n} \to f_{i+1,n+2^i} \\ g_{i,n} \to g_{i+1,n+2^i} \end{cases}$$

- 转移：

$$
\begin{cases}
f_{i,n} \times \binom{2^i}{r} \to f_{i+1,n+r} \\
f_{i,n} \times \binom{2^i-1}{r} \to g_{i+1,n+r} \\
f_{i,n} \to f_{i+1,n+2^i} \\
g_{i,n} \to f_{i+1,n+2^i} \\
g_{i,n} \to g_{i+1,n+2^i}
\end{cases}
$$

- 使用卷积优化，则枚举 $p$ 后 DP 的时间复杂度为 $\sum_i O(i2^i) = O(k2^k)$。

- 转移：

$$\begin{cases} f_{i,n} \times \binom{2^i}{r} \to f_{i+1,n+r} \\ f_{i,n} \times \binom{2^i-1}{r} \to g_{i+1,n+r} \\ f_{i,n} \to f_{i+1,n+2^i} \\ g_{i,n} \to f_{i+1,n+2^i} \\ g_{i,n} \to g_{i+1,n+2^i} \end{cases}$$

- 使用卷积优化，则枚举 $p$ 后 DP 的时间复杂度为 $\sum_i O(i2^i) = O(k2^k)$。
- 时间复杂度 $O(k^2 2^k + T)$，空间复杂度 $O(k^2 2^k)$。

**问题 (LOJ 4080)**

给定 $1 \sim n$ 的排列 $p$，求将 $p$ 划分为 $q, r$ 后，$q$ 的前缀最大值个数与 $r$ 的前缀最小值个数之和的最大值。

数据范围：$2 \le n \le 2 \times 10^5$。

- 考虑按下标顺序 DP：

- 考虑按下标顺序 DP：
- 设 $f_{i,a,b}$ 表示划分了 $p$ 的前 $i$ 个元素，$q$ 中的最大值为 $a$，$r$ 中的最小值为 $b$ 的答案。

- 考虑按下标顺序 DP：
- 设 $f_{i,a,b}$ 表示划分了 $p$ 的前 $i$ 个元素，$q$ 中的最大值为 $a$，$r$ 中的最小值为 $b$ 的答案。
- 转移可以根据 $p_i$ 放在 $q$ 或 $r$ 中 $O(1)$ 分类讨论得出。

- 注意到当 $a > b$ 时，$i$ 之后的元素都可以选择放入 $q$ 或 $r$ 中，且不改变相关的最值，于是之后的贡献一定为首项 $> a$ 的最长上升子序列长度与首项 $< b$ 的最长下降子序列长度之和，这可以通过预处理后用线段树维护快速求出。

- 注意到当 $a > b$ 时，$i$ 之后的元素都可以选择放入 $q$ 或 $r$ 中，且不改变相关的最值，于是之后的贡献一定为首项 $> a$ 的最长上升子序列长度与首项 $< b$ 的最长下降子序列长度之和，这可以通过预处理后用线段树维护快速求出。
- 当 $a < b$ 时，前 $i$ 个元素中不可能包含 $(a, b)$ 中的元素，否则一定会影响某一个序列的最值，于是对于每个 $i$，可能的 $a, b$ 只有 $O(n)$ 种。

- 注意到当 $a > b$ 时，$i$ 之后的元素都可以选择放入 $q$ 或 $r$ 中，且不改变相关的最值，于是之后的贡献一定为首项 $> a$ 的最长上升子序列长度与首项 $< b$ 的最长下降子序列长度之和，这可以通过预处理后用线段树维护快速求出。
- 当 $a < b$ 时，前 $i$ 个元素中不可能包含 $(a, b)$ 中的元素，否则一定会影响某一个序列的最值，于是对于每个 $i$，可能的 $a, b$ 只有 $O(n)$ 种。
- 利用上述性质优化 DP 可以做到 $O(n^2 \log n)$ 或 $O(n^2)$ 的时间复杂度。

- 考虑整体 DP，对 $a$ 一维建立线段树记录 $f_i$ 的 DP 值。

- 考虑整体 DP，对 $a$ 一维建立线段树记录 $f_i$ 的 DP 值。
- 从 $f_i$ 转移到 $f_{i+1}$ 时，DP 值只需要进行单点修改。

- 考虑整体 DP，对 $a$ 一维建立线段树记录 $f_i$ 的 DP 值。
- 从 $f_i$ 转移到 $f_{i+1}$ 时，DP 值只需要进行单点修改。
- 考虑转移到 $a > b$ 的状态的贡献，需要用 DP 值与 LIS 与 LDS 之和更新答案，于是可以直接将和记录在线段树中。

- 考虑整体 DP，对 $a$ 一维建立线段树记录 $f_i$ 的 DP 值。
- 从 $f_i$ 转移到 $f_{i+1}$ 时，DP 值只需要进行单点修改。
- 考虑转移到 $a > b$ 的状态的贡献，需要用 DP 值与 LIS 与 LDS 之和更新答案，于是可以直接将和记录在线段树中。
- LIS 与 LDS 的修改为区间加形式，更新答案只需要区间查询最大值。

- 考虑整体 DP，对 $a$ 一维建立线段树记录 $f_i$ 的 DP 值。
- 从 $f_i$ 转移到 $f_{i+1}$ 时，DP 值只需要进行单点修改。
- 考虑转移到 $a > b$ 的状态的贡献，需要用 DP 值与 LIS 与 LDS 之和更新答案，于是可以直接将和记录在线段树中。
- LIS 与 LDS 的修改为区间加形式，更新答案只需要区间查询最大值。
- 时间复杂度 $O(n \log n)$，空间复杂度 $O(n)$。