另一个杂题选讲

4182_543_731

24/02/18

Inversion Sum(agc030d)

给一个 n 阶排列 p。进行 q 次操作，每次给定 x_i, y_i，然后可能

- 交换 p_{x_i} 和 p_{y_i}，或者
- 不进行任何操作

对所有 2^q 种情况，求和最终 p 的逆序对数量。

$n, q \leq 3000$

$3s/1024MB$

Inversion Sum(agc030d)

从整体考虑排列是困难的，因为可能的情况有 $n!$ 种。

但计算逆序对的贡献是容易被拆开的：$inv(p) = \sum_{i<j}[p_i > p_j]$。那么根据期望线性性，我们也可以对于每对 $i < j$ 求出使得 $p_i > p_j$ 的方案数，求和即可得到答案。

Inversion Sum(agc030d)

从整体考虑排列是困难的，因为可能的情况有 $n!$ 种。

但计算逆序对的贡献是容易被拆开的：$inv(p) = \sum_{i<j}[p_i > p_j]$。那么根据期望线性性，我们也可以对于每对 $i < j$ 求出使得 $p_i > p_j$ 的方案数，求和即可得到答案。

更进一步，因为操作都是交换，交换后两个位置的大小关系也是交换前某两个位置的大小关系。从而我们在整个过程中都只需要维护两个位置大小关系的情况。

Inversion Sum(agc030d)

具体来说，设 $dp_{k,i,j}$ 表示 k 次操作后使得 $p_i > p_j$ 的方案数。考虑上一次操作容易得到转移：

- 如果不进行交换，则直接从 $dp_{k-1,i,j}$ 转移。
- 如果进行交换，那么从 dp_{k-1,ls_i,ls_j} 转移。其中 ls_i 表示位置 i 的数这次交换前的位置。

Inversion Sum(agc030d)

具体来说，设 $dp_{k,i,j}$ 表示 k 次操作后使得 $p_i > p_j$ 的方案数。考虑上一次操作容易得到转移：

- 如果不进行交换，则直接从 $dp_{k-1,i,j}$ 转移。
- 如果进行交换，那么从 dp_{k-1,ls_i,ls_j} 转移。其中 ls_i 表示位置 i 的数这次交换前的位置。

容易发现交换只影响 $O(n)$ 对位置，剩下的位置都是直接 $\times 2$。那么我们可以原地操作，每次更新这 $O(n)$ 个位置，整体乘打标记处理。另一种角度是看成求期望，两者实现上等价。复杂度 $O(n^2 + nq)$

takeaway: 线性性可以用来分解一些复杂的贡献计数，将其变为若干简单的问题。

Range Set(agc045c)

给定 n, a, b。有一个长度为 n，初始全 0 的字符串。可以对其进行任意次如下操作：

- 将连续的 a 个字符变为 0。
- 将连续的 b 个字符变为 1。

求可以达到的字符串种数。

$n \leq 5000$

$2s/1024MB$

Range Set(agc045c)

对于各种数方案数题，首先应该考虑如何判定一种方案是否能被达到（如果这不是显然的）。
具体到这里，考虑一个终止 01 串状态，我们需要判定它是否能被初始状态达到。
之后的操作会覆盖之前的改变，使得正着考虑较为困难。

Range Set(agc045c)

对于各种数方案数题，首先应该考虑如何判定一种方案是否能被达到（如果这不是显然的）。
具体到这里，考虑一个终止 01 串状态，我们需要判定它是否能被初始状态达到。
之后的操作会覆盖之前的改变，使得正着考虑较为困难。因此考虑倒过来看：如果最后有连续的 a 个 0 或者 b 个 1，就可以在这里有一次操作，操作前这些可以是任意值。
经典地，考虑倒回去时将这些位置变为?，在需要时再变成对应的值。那么倒过来的操作可以看成

- 选择连续的 a 个均为 0 或? 的字符变为全?。
- 选择连续的 b 个均为 1 或? 的字符变为全?。

我们需要回到全 0，即只存在 0 和?。但此时可以再进行第一种操作，因此可以要求回到全?。
此时目标只要?，可以看出 $0,1$ 对称，从而我们可以交换 a,b。不妨设 $a \leq b$。

Range Set(agc045c)

继续分析合法性的问题。可以发现只要我们进行了第二种操作造出连续 b 个?，接下来就可以每次拿 $b-1$ 或 $a-1$ 个? 和旁边一个字符操作达成目标，即只要进行了一次 b 的操作就一定合法，同时我们也必须这样操作消掉 1（全 0 是特殊情况，但可以发现没问题）。
为了做一次 1 的操作，我们需要在只用 0 的操作的情况下，尽量把 0 变成?。每段被 1 分隔开的 0 是独立的，只要有 a 个 0 就可以处理这一段。从而可以发现合法条件为：

将每一个长度至少是 a 的连续 0 段变成 1 后，存在连续 b 个 1。

证明只需要沿着上面的分析。

Range Set(agc045c)

最后的计数是容易的，这里随机给一种方式：

考虑数不合法的方案数。相当于长度小于 a 的 0 段把整个序列分成了若干长度小于 b 的部分。

剩余每部分中每一段 0 长度都至少是 a，且和分界点相邻的位置必须是 1。考虑先求出 f_n 表示长度为 n 的满足上述条件的序列数。不合法方案一定是这样的段和小于 a 的 0 段交替。直接 dp 记录当前总长，每次放下一段。注意边界。

复杂度 $O(n^2)$

takeaway: 正难则反，避免局限于一个角度。

Avoid Permutations(arc118e)

使用如下方式定义一个 n 阶排列 p 的权值:

考虑一个 $(n+2)\times(n+2)$ 的网格，其中行列均标号为 $0,1,\cdots,n+1$。考虑从 $(0,0)$ 走到 $(n+1,n+1)$，每一步可以让两维坐标中的一个加一。排列 p 的权值为满足上述条件且不经过任意一个 (i,p_i) 位置的方案数。

现在给出排列部分位置的值，考虑所有剩余部分任意排列得到的所有排列，求和它们的权值。

$n\leq 200$

$2s/1024MB$

Avoid Permutations(arc118e)

问题相当于选一个排列，再选一条与排列不交的路径。即相当于统计有多少对排列和路径不交。

经典地考虑容斥：枚举一些交点，系数是 $(-1)^C$ 乘上方案数，也就是经过所有交点的排列数乘上经过所有交点的路径数。首先分别考虑两者如何计算。

Avoid Permutations(arc118e)

问题相当于选一个排列，再选一条与排列不交的路径。即相当于统计有多少对排列和路径不交。

经典地考虑容斥：枚举一些交点，系数是 $(-1)^C$ 乘上方案数，也就是经过所有交点的排列数乘上经过所有交点的路径数。首先分别考虑两者如何计算。

对于排列，每个点 (i,j) 限制 $p_i = j$。显然两个限制相矛盾当且仅当它们下标或值相同，即有一行或一列选了两个不同格子。如果不矛盾则方案数为剩余行数的阶乘。

Avoid Permutations(arc118e)

问题相当于选一个排列，再选一条与排列不交的路径。即相当于统计有多少对排列和路径不交。

经典地考虑容斥：枚举一些交点，系数是 $(-1)^C$ 乘上方案数，也就是经过所有交点的排列数乘上经过所有交点的路径数。首先分别考虑两者如何计算。

对于排列，每个点 (i,j) 限制 $p_i = j$。显然两个限制相矛盾当且仅当它们下标或值相同，即有一行或一列选了两个不同格子。如果不矛盾则方案数为剩余行数的阶乘。

对于路径，因为只能向 $+1$ 方向走，存在路径经过所有选中点当且仅当可以将所有点排成两维坐标均不降的一列（由上一部分可知可以限制递增），方案数就是从每个点走到下个点的方案数乘积。

排成坐标递增的一列后，排列部分不矛盾的限制也只剩下和给定点不矛盾。但如果做直接枚举下个点的 dp，复杂度可能高达 $O(n^5)$。

Avoid Permutations(arc118e)

排列很难直接算，但路径是容易统计的：DP 每次走下一步。

从路径的角度考虑，相当于选一条路径，路径上再选若干点，系数为 $(-1)^C$ 乘上排列方案数。通过之前对排列的讨论可以发现：

- 某些导致矛盾的位置不能选。
- 同行同列不能选多个位置。
- 需要记录（给定和选点同时）空余的行数。

这在路径的 dp 上都容易维护：因为只向 $+1$ 的方向走，只需要记录当前行列是否已经被选过，之前有多少空行。

复杂度 $O(n^3)$

loj3646

一棵 n 个点的树，点带权 a_i。

q 次询问，每次给 u, v, k，求从 u 到 v 路径上选出 k 个点，将它们的点权按位 AND 可以得到的最大结果。强制在线。

$n \leq 10^6, q \leq 10^5, k \leq 10, a_i < 2^{62}$

$6s/512MB$

loj3646

树上路径问题的常见想法是合并信息，但这里会遇到很大的困难，因为给若干个数的子问题就不好做。

先考虑这一子问题的做法。

loj3646

树上路径问题的常见想法是合并信息，但这里会遇到很大的困难，因为给若干个数的子问题就不好做。

先考虑这一子问题的做法。显然高位比低位重要，可以发现如下贪心。

从高到低依次考虑每一位，此时答案这一位取 1 一定优于取 0，从而：

- 如果至少有 k 个数在这一位为 1，则这一位必定取 1，因此可以删去所有这一位为 0 的数。
- 否则，满足高位的情况下这一位不可能取到 1，因此可以跳过这一位。

这还是没法上树合并。但这一问题看起来只有这一种做法，因此我们只能尝试继续分析它的性质。

loj3646

尝试通过各种角度考虑这个算法。可以发现如果先对整个序列排序，整个过程可以得到简化：判定这一位是否取 1 的过程只需要看最大的 k 个数的最高位，然后

- 如果可以，那么移除最高位的同时删掉了这一位不是 1 的数，因此留下的部分大小顺序不变。
- 否则，移除最高位的时候只会影响不超过 $k-1$ 个数，剩余数大小关系不变。

...?

loj3646

尝试通过各种角度考虑这个算法。可以发现如果先对整个序列排序，整个过程可以得到简化：判定这一位是否取 1 的过程只需要看最大的 k 个数的最高位，然后

- 如果可以，那么移除最高位的同时删掉了这一位不是 1 的数，因此留下的部分大小顺序不变。
- 否则，移除最高位的时候只会影响不超过 $k-1$ 个数，剩余数大小关系不变。

...?

每次我们只看最大 k 个数，最多 pop_front $k-1$ 个数。那么

Lemma

我们只会使用前 $(k-1)\log V+1$ 大的数。

loj3646

现在只需要查链上前 k 大。这是经典数据结构问题，下面随机给一个实现。

经典的做法是每次拿出剩下的最大的。先树剖出 $O(\log^2 n)$ 或者 $O(\log n)$ 个区间，每个区间记录一个最大值。然后每次找到最大值所在区间，从最大值将区间分成两部分放回去。

如果用 ST 表找区间 max，这里复杂度是 $O((k\log V+\log n)\log(k\log V+\log n))$

对于前面那部分，如果用优先队列实现则复杂度 $O(k\log V\log(k\log V))$，可以通过。也有更暴力的通过方式

takeaway: 一些简单但不平凡，甚至本质的子问题是值得仔细思考的。

Bus(CTT23 d1t1)

数轴上有 n 个物品，第 i 个物品初始在 s_i，你需要把它放到 t_i。

你在数轴上移动，起点给定。你可以在经过物品时拿起它，在任意位置放下任意物品，同时你可以同时拿任意多个物品。

求从起点开始移动，将所有物品放到需要位置的最小移动距离。

更进一步，给定 q 次询问，每次指定起点，然后你需要回答这个起点的答案。

$n, q \leq 10^6$

subtask1: $q = 1$，起点在所有 s_i 和 t_i 左侧

subtask2: $q = 1$

$2s/1024MB$

Bus(CTT23 d1t1)

在出题人/验题人良心的情况下，特殊的部分分通常可以作为切入点。

考虑最简单的情况。称所有 s_i, t_i 的最小值为左端点，最大值为右端点。此时可以看成从左端点开始。

显然我们一定会经过右端点。那么从左端点走到右端点的这个过程一定可以处理所有向右放 $(t_i \geq s_i)$ 的物品。我们只需要考虑处理向左放的物品，此时需要倒过来走覆盖这些 $[t, s]$ 的区间。

一次倒着走的过程只和倒着走的距离有关，中间折返没用且不优，从而只有两种方式。

1. 在中间折返覆盖一段，额外距离为 $2l$。
2. 在走到右端点后回来覆盖一段后缀，额外距离为 l。

Bus(CTT23 d1t1)

1. 在中间折返覆盖一段，额外距离为 $2l$。
2. 在走到右端点后回来覆盖一段后缀，额外距离为 l。

先考虑一种简化情况：没有后缀的操作。此时只需要考虑每一段的总长。

我们需要用最少总长覆盖每一段 $[t_i, s_i]$，显然一种方式是覆盖它们的并：对于并的每一个区间用一次折返覆盖。

Bus(CTT23 d1t1)

1. 在中间折返覆盖一段，额外距离为 $2l$。
2. 在走到右端点后回来覆盖一段后缀，额外距离为 l。

先考虑一种简化情况：没有后缀的操作。此时只需要考虑每一段的总长。

我们需要用最少总长覆盖每一段 $[t_i, s_i]$，显然一种方式是覆盖它们的并：对于并的每一个区间用一次折返覆盖。

对于后缀操作，考虑直接枚举后缀覆盖了多少。先用若干个不交线段表示上面的并，然后一定是覆盖后面若干条线段。这个过程容易 $O(n)$，也可以扫过去直接求出。

Bus(CTT23 d1t1)

然后考虑起点任意的问题。根据部分分的提示，考虑将问题转化到类似之前的情况。
之前的特点是我们从左端点走到了右端点。可以发现现在仍然有类似的情况：我们必定经过两个端点，因此一定是先经过左端点再经过右端点。两种情况对称，做两遍就可以处理。这里只分析第一种。
此时情况和之前非常相似，唯一区别在于还有开头从起点走到左端点的一段路。

Bus(CTT23 d1t1)

然后考虑起点任意的问题。根据部分分的提示，考虑将问题转化到类似之前的情况。

之前的特点是我们从左端点走到了右端点。可以发现现在仍然有类似的情况：我们必定经过两个端点，因此一定是先经过左端点再经过右端点。两种情况对称，做两遍就可以处理。这里只分析第一种。

此时情况和之前非常相似，唯一区别在于还有开头从起点走到左端点的一段路。

左端点到右端点可以解决所有向右的，这里就只需要考虑解决向左的。可以发现只需要考虑向右走到一个 r 然后转向的情况：因为会走到左端点，能覆盖的只和能到的最大 r 有关，那上面这样是最优的。

考虑枚举转向位置 r，它能覆盖所有 $s_i \leq r$ 的位置，然后我们需要求出只考虑剩余区间时上一页问题的答案。

Bus(CTT23 d1t1)

考虑从大到小枚举 r，那么会不断地往问题左侧加入线段。从线段并的角度，整个过程一直在修改或者加入第一段。

有很多种方式让之前的问题支持这样修改。

Bus(CTT23 d1t1)

考虑从大到小枚举 r，那么会不断地往问题左侧加入线段。从线段并的角度，整个过程一直在修改或者加入第一段。

有很多种方式让之前的问题支持这样修改。例如：从右往左做，记 dp_i 表示只考虑第 i 段及之后段时的答案，然后有 $dp_i = \min(R - r_i, dp_{i+1} + 2(r_i - l_i)$。这就解决了单组询问。

对于多组询问，注意到上述过程可以求出每个 r 转向后的路程，只需要再从右到左扫一次就可以求出每个起点的答案。

复杂度 $O((n+q)\log(n+q))$

takeaway: 对于过于复杂的问题，可以考虑从简单情况开始。

Message(qoj6366)

有一个长度为 n 的字符串 S，你可以进行如下操作：

选择一种字符，删除这种字符的第一次出现或最后一次出现。

删除字符会产生代价。删除**初始时**在位置 i 的字符需要 c_i 的代价。

给定目标串 T，求操作得到 T 的最小代价，或输出无解。

$n \leq 2 \times 10^5, |\sum| = 26$

$2s/256MB$

Message(qoj6366)

这个操作可以被描述为，对每种字符保留它的所有出现中的一个区间。但我们显然难以同时决定 $|\sum|$ 个区间的位置。

这里等于 T 是很强的限制，考虑从这个限制出发，看能不能更好地处理选择区间端点的问题。

Message(qoj6366)

这个操作可以被描述为，对每种字符保留它的所有出现中的一个区间。但我们显然难以同时决定 $|\sum|$ 个区间的位置。

这里等于 T 是很强的限制，考虑从这个限制出发，看能不能更好地处理选择区间端点的问题。

考虑 S 中每种字符保留区间的端点。在保留这些字符到 T 的过程中，相对顺序不会发生改变，从而 T 中这些端点顺序不变。那么考虑在 T 中找到每种字符出现区间的端点，这些端点的顺序可以还原所有保留区间的端点顺序。

这些端点可以把 S 分成若干段，也把 T 分成了若干段，考虑从这里分析。

Message(qoj6366)

考虑一组解，按照保留区间端点将 S 分成 $2|\sum|+1$ 段。通过端点顺序可以确定第 i 段内保留的字符种类，通过在 T 中分段，我们还可以确定这一段保留下来的结果必须等于 T 中的这一段。

Message(qoj6366)

考虑一组解，按照保留区间端点将 S 分成 $2|\sum|+1$ 段。通过端点顺序可以确定第 i 段内保留的字符种类，通过在 T 中分段，我们还可以确定这一段保留下来的结果必须等于 T 中的这一段。

那么问题变为，将 S 分成 $2|\sum|+1$ 段，使得第 i 段在保留集合 C_i 中的字符后等于 t_i。

然后是简单 dp：依次考虑每一段，设 $dp_{k,i}$ 表示前 k 段划分到 i 的最小代价。考虑下一段时，先只保留全串中 C_i 内的字符，然后就可以通过匹配知道从每个 i 开始能不能合法地分出下一段，然后能转移到一段空位。转移是容易的。

复杂度 $O(n|\sum|)$

Message(qoj6366)

考虑一组解，按照保留区间端点将 S 分成 $2|\sum|+1$ 段。通过端点顺序可以确定第 i 段内保留的字符种类，通过在 T 中分段，我们还可以确定这一段保留下来的结果必须等于 T 中的这一段。

那么问题变为，将 S 分成 $2|\sum|+1$ 段，使得第 i 段在保留集合 C_i 中的字符后等于 t_i。

然后是简单 dp：依次考虑每一段，设 $dp_{k,i}$ 表示前 k 段划分到 i 的最小代价。考虑下一段时，先只保留全串中 C_i 内的字符，然后就可以通过匹配知道从每个 i 开始能不能合法地分出下一段，然后能转移到一段空位。转移是容易的。

复杂度 $O(n|\sum|)$

另一种思路是将限制分解为每对字符种类间的限制，然后可以发现如果两种字符在 T 上区间相交，则确定一种的区间就可以唯一确定另一种，否则可以限制另一种端点的大小。那么所有相交的可以一起确定，剩下若干相离段再做 dp。

Synchronized Subsequence(agc026e)

给一个长度为 $2n$，包含 n 个 0 和 n 个 1 的字符串。你可以选择它的任意子序列，满足如下限制：

$\forall i$，原字符串中的第 i 个 0 和第 i 个 1 必须同时被选或者同时不选。

求你能得到的字典序最大的字符串。

$n \leq 3000$

bonus: $n \leq 2 \times 10^5$

$2s/1024MB$

Synchronized Subsequence(agc026e)

相当于有若干对 01 或者 10，每一对必须同时选。一大问题出在不同对之间可能相交。先考虑它们之间的位置关系。

如果相邻两段不交，则这两对间存在分界线使得左侧 01 数量相等。换言之，考虑将 1 看成 $+1$，0 看成 -1 求前缀和，那么所有前缀和为 0 的位置可以将序列分成若干段，不同段之间的选择是独立的。

我们仍然从简单情况开始：考虑单个段的情况。此时有两种不同的情况：段内每一个 1 都早于对应 0 出现（对应前缀和 > 0），或者每个 0 都早于对应 1。

Synchronized Subsequence(agc026e)

相当于有若干对 01 或者 10，每一对必须同时选。一大问题出在不同对之间可能相交。先考虑它们之间的位置关系。

如果相邻两段不交，则这两对间存在分界线使得左侧 01 数量相等。换言之，考虑将 1 看成 $+1$，0 看成 -1 求前缀和，那么所有前缀和为 0 的位置可以将序列分成若干段，不同段之间的选择是独立的。

我们仍然从简单情况开始：考虑单个段的情况。此时有两种不同的情况：段内每一个 1 都早于对应 0 出现（对应前缀和 > 0），或者每个 0 都早于对应 1。

第二种情况是简单的：我们只能选出若干 01，那么显然多对相交得到的 0011 或者更多 0 比 01 更差。即如果存在多对相交，删掉它们只保留一对更优。因而有如下结论：

如果选了一对 01，则不会选任何和它相交的 01。

因此答案形如 $010101\cdots$，可以贪心求：每次选能选的第一对。

Synchronized Subsequence(agc026e)

考虑第一种情况，此时正好与之前相反：相交得到的 1100 比 10 更优。考虑能否类似得到最优解满足的性质。考虑相交两对可以发现：

考虑两对相对位置形如 1100 的 10。如果前一对选了则后一对必选。

证明：考虑加入后一对 10，它会在某一段 1 里面放一个 1，但还会在后面放一个 0。但当前 1 段一定在第一对的 0 或之前就被截断了，因此这确实让这一段 1 的长度增加了，从而更优。

(反方向是不成立的，这正是因为新加入的 0 可能截断当前 1 段)

因为每一段内相邻两对总是相交，选一对就会导致选后面的所有对。因此当前段一定是选择了后若干对。

Synchronized Subsequence(agc026e)

考虑第一种情况，此时正好与之前相反：相交得到的 1100 比 10 更优。考虑能否类似得到最优解满足的性质。考虑相交两对可以发现：

考虑两对相对位置形如 1100 的 10。如果前一对选了则后一对必选。

证明：考虑加入后一对 10，它会在某一段 1 里面放一个 1，但还会在后面放一个 0。但当前 1 段一定在第一对的 0 或之前就被截断了，因此这确实让这一段 1 的长度增加了，从而更优。

(反方向是不成立的，这正是因为新加入的 0 可能截断当前 1 段)

因为每一段内相邻两对总是相交，选一对就会导致选后面的所有对。因此当前段一定是选择了后若干对。

此时已经可以做到 $O(n^2)$ 了：设 dp_i 表示考虑后 i 对的最优解，每次枚举当前对选不选向后选/跳过若干段到一个和前面独立的地方继续。但我们还能更优。

Synchronized Subsequence(agc026e)

考虑不同段之间的选择。显然 10 段（第一种情况）选出来的都比 01 段优，因此往后还有 10 段时我们不会选择 01 段。因此我们可以先考虑简单情况：只有 10 段。

现在一段有 $O(n)$ 种可能的选择。虽然要求字典序最大可以去掉不优的情况，但如果一个是另一个的前缀这就不能直接确定。

Synchronized Subsequence(agc026e)

考虑不同段之间的选择。显然 10 段（第一种情况）选出来的都比 01 段优，因此往后还有 10 段时我们不会选择 01 段。因此我们可以先考虑简单情况：只有 10 段。

现在一段有 $O(n)$ 种可能的选择。虽然要求字典序最大可以去掉不优的情况，但如果一个是另一个的前缀这就不能直接确定。

但这里问题有很好的形式：每种选择都是选后缀一串依次相交的 10 对，那么用前缀和的角度整个过程只在开头结尾前缀和为 0。因此不可能有前缀的情况，每一段内方案唯一。每种方案通过第一个 0 的位置看成一串 1 接一个后缀，后缀排序即可求出最优解。

Synchronized Subsequence(agc026e)

最后问题变为有若干个串，选一些按原顺序拼起来求最大字典序。

考虑选下一段的过程。类似之前的结论，这里不会出现一个是另一个前缀的情况（除了相等）。那么可以直接贪心选后面字典序最大的串，相同时可以发现选前面更优（接下来选择更多）。因此每次贪心这样向后选即可。这里也可以后缀排序实现。

01 段只会在最后选，此时变成之前只有 01 的情况。

然后就可以做到“$O(n)$ 甚至 $O(n\log n)$”

Paired Wizards(arc142f)

有两个数 x, y，其初始均为 0。你还有一个分数，初始为 0。

有两种操作 $0, 1$。对一个数进行操作 0 会让它加一，对其进行操作 1 会让你的分数加上当前数的值。

现在有 q 轮操作顺序发生，每轮你对两个数分别进行一次操作，但每一轮你只能在该轮给定的两种行动中选一种：

- 对 x 进行操作 a_i，对 y 进行操作 b_i。或者
- 对 x 进行操作 c_i，对 y 进行操作 d_i。

求最后的最大分数。

$n \leq 8000$

$3s/1024MB$

Paired Wizards(arc142f)

每轮操作有如下几种可能：

- 对两个数的操作都固定。
- 对 x 的操作任意，对 y 的操作固定。这种情况记作 x。对称的另一种情况记作 y。
- 对两个数的操作必须相同，记作 $00/11$。
- 对两个数操作相反，记作 $01/10$。

这里的分数也可以拆开：看成每个数中前面的 0 操作和后面的 1 操作组成的对数。因此整个贡献可以分为四类情况内部（以及和固定部分）的贡献和它们相互的贡献。

考虑从简单情况开始：先分析只有一种情况（和一些固定位置）的问题。

Paired Wizards(arc142f)

先考虑只有 x 操作的情况。此时可以看成只有 x，有一些操作已经确定，需要决定另一些操作。考虑最优解需要满足的性质。可以发现：

Paired Wizards(arc142f)

先考虑只有 x 操作的情况。此时可以看成只有 x，有一些操作已经确定，需要决定另一些操作。考虑最优解需要满足的性质。可以发现：

最优解一定在未确定操作中先选 0 再选 1。

证明：如果有一对 10，换过来严格更优。这在有别的操作的时候也成立。

Paired Wizards(arc142f)

先考虑只有 x 操作的情况。此时可以看成只有 x，有一些操作已经确定，需要决定另一些操作。考虑最优解需要满足的性质。可以发现：

最优解一定在未确定操作中先选 0 再选 1。

证明：如果有一对 10，换过来严格更优。这在有别的操作的时候也成立。

进一步分析，可以看出这东西非常凸，考虑比较选 $i-1$ 个和 i 个 0 的方案，即固定前面都是 0，后面都是 1，考虑第 i 个位置选啥更优。

如果它选 0 则贡献是右侧 1 的数量，选 1 则贡献是左侧 0 的数量。前者更优当且仅当这个差大于 0，即

$$(\text{右侧 1 数量}) - (\text{左侧 0 数量}) \geq 0$$

$$(\text{右侧操作数}) - (\text{左侧 0 数量}) - (\text{右侧 0 数量}) \geq 0$$

$$(\text{右侧操作数}) - (i-1) - (\text{固定的 0 数量}) \geq 0$$

Paired Wizards(arc142f)

$$(\text{右侧操作数}) - (i-1) \geq (\text{固定的 0 数量})$$

左侧是关于 i 递减的，因此存在一个分界点，左侧多选 0 更优，右侧少选 0 更优。这个分界点就是最优解。

然后考虑答案，根据分界点的分析我们可以从全 1 开始往开头加 0 直到分数增量小于 0，那么答案是

$$(\text{全选 1 的答案}) + \sum_i \max(0, (\text{右侧操作数}) - (i-1) - (\text{固定的 0 数量}))$$

这些分析对 y 操作和 $00/11$ 操作也显然成立，只是在后者时需要同时考虑两种数上的操作。

Paired Wizards(arc142f)

考虑只有 $01/10$ 操作的情况。仍然考虑调整，将一个位置从 10 换成 01 会使分数增加

$$\begin{aligned}&(\text{右侧 x 的 1 操作数})+(\text{左侧 y 的 0 操作数})-(\text{左侧 x 的 0 操作数})-(\text{右侧 y 的 1 操作数})\\=&(\text{右侧 x 的 1 操作数})+(\text{左侧 x 的 1 操作数})-(\text{左侧 y 的 1 操作数})-(\text{右侧 y 的 1 操作数})\end{aligned}$$

那这只和两个数 1 操作数量差（等价于 0 操作数量差）有关，从而整个过程只和换的个数有关，和位置都无关。换第 i 个时的增量是

$$(\text{x 的固定 1 操作数})-(\text{y 的固定 1 操作数})+(s-i)-(i-1)$$

其中 s 是总的 $01/10$ 操作数量。因此这还是可以分界点，贪心一个一个翻。

Paired Wizards(arc142f)

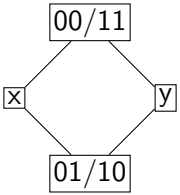
现在我们将不同种类的操作放在一起，考虑不同种类的选择之间的影响。

显然 x 操作和 y 操作完全不相关。

Paired Wizards(arc142f)

现在我们将不同种类的操作放在一起，考虑不同种类的选择之间的影响。

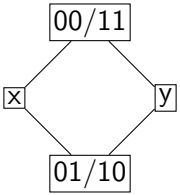
显然 x 操作和 y 操作完全不相关。进一步可以发现，一个 $01/10$ 和一个 $00/11$ 操作之间的贡献和 $01/10$ 的选择是无关的，因此这部分贡献可以放到 $00/11$ 里面去考虑。即如下关系



Paired Wizards(arc142f)

现在我们将不同种类的操作放在一起，考虑不同种类的选择之间的影响。

显然 x 操作和 y 操作完全不相关。进一步可以发现，一个 $01/10$ 和一个 $00/11$ 操作之间的贡献和 $01/10$ 的选择是无关的，因此这部分贡献可以放到 $00/11$ 里面去考虑。即如下关系



此时可以考虑枚举 x, y 操作的 $O(n^2)$ 种情况，再独立地处理另外两种情况。具体地，可以先假设一个初始状态，然后枚举 x, y，再对另外两种分别调整到最优解。此时后面的两步都是之前的情况。

Paired Wizards(arc142f)

考虑 $00/11$ 的部分。求这部分最优解的一种方式是从全 11 开始逐渐将开头换成 00。之前已经分析过，这个过程的增量递减从而可以贪心，最终增量为

$$\sum_i \max(0, (\text{右侧操作数}) - 2(i-1) - (\text{固定的 0 数量}))$$

可以发现这是一个只关于整体固定 (除去 $00/11$ 操作外) 的 0 数量的上凸函数，可以 $O(n)$ 预处理求出每种数量的答案。

Paired Wizards(arc142f)

考虑 $00/11$ 的部分。求这部分最优解的一种方式是从全 11 开始逐渐将开头换成 00。之前已经分析过，这个过程的增量递减从而可以贪心，最终增量为

$$\sum_i \max(0, (\text{右侧操作数}) - 2(i-1) - (\text{固定的 0 数量}))$$

可以发现这是一个只关于整体固定 (除去 $00/11$ 操作外) 的 0 数量的上凸函数，可以 $O(n)$ 预处理求出每种数量的答案。

然后考虑另一侧 $01/10$，类似的写出增量为

$$\sum_i \max(0, (\text{x 的固定 1 操作数}) - (\text{y 的固定 1 操作数}) + s - 2i + 1)$$

这是关于两个数 0 操作数量差的上凸函数，也可以 $O(n)$ 求。

再预处理 x, y 操作每种选择的贡献，完全预处理后枚举即可做到 $O(n^2)$。

Tom & Jerry(loj3406)

给一张 n 个点 m 条边的无向连通简单图，Alice 和 Bob 在图上进行如下博弈：

两人轮流行动，Alice 先手。Alice 每次可以经过任意条边（可以不走）但不能经过 Bob 当前位置。Bob 每次最多走一条边，Bob 抓到 Alice 则 Bob 获胜。如果经过 n^2 轮游戏还未结束则 Alice 获胜。

q 次询问，每次给定双方初始位置，求谁获胜。

$n, m, q \leq 10^5$

$1s/512MB$

Tom & Jerry(loj3406)

对博弈进行一些分析。每一轮 Alice 可以走除了 Bob 当前所在点外的所有点，因此他能到的点是删去 Bob 所在点后当前点所在的连通块。

删一个点的连通性显然对应点双。因此考虑建出点双的圆方树，那么根据割点的简单性质，Alice 每次能到的点就是圆方树上不经过 Bob 能到的点。

但这还是很复杂。沿着上面的思路，我们从最简单的情况开始：图是单个点双。此时 Alice 每步可以走到任意点，但不能被 Bob 一步抓到。可以发现：

Tom & Jerry(loj3406)

对博弈进行一些分析。每一轮 Alice 可以走除了 Bob 当前所在点外的所有点，因此他能到的点是删去 Bob 所在点后当前点所在的连通块。

删一个点的连通性显然对应点双。因此考虑建出点双的圆方树，那么根据割点的简单性质，Alice 每次能到的点就是圆方树上不经过 Bob 能到的点。

但这还是很复杂。沿着上面的思路，我们从最简单的情况开始：图是单个点双。此时 Alice 每步可以走到任意点，但不能被 Bob 一步抓到。可以发现：

在一个点双内，如果存在点连向所有点则 Bob 胜，否则 Alice 胜。

证明：Bob 走到这个点显然直接获胜，否则 Alice 每步只需要走到 Bob 不能一步到的点。

称这样的点双是好的。

Tom & Jerry(loj3406)

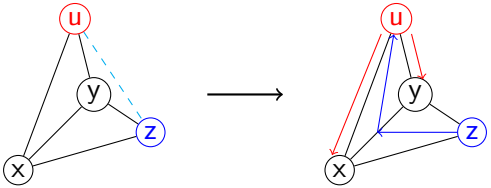
回到圆方树上考虑。如果 Alice 能直接走到一个这样的点双，那走过去就直接赢了：如果有外面的点连向点双内所有点，那它也应该加入点双。

否则，Alice 就不能停在一个点双里面，因此我们需要回到圆方树上考虑问题。

如果这是一棵普通的树，那 Bob 只需要不断向 Alice 所在的子树走就能抓住。因此这里 Alice 想要不被抓住必须逃出子树：在 Bob 从上面进入一个点双时离开当前子树。考虑什么情况下这是可行的。

Tom & Jerry(loj3406)

类似之前的思路可以发现：如果 Bob 进入点双时所在的点不连向点双内所有点，则 Alice 就可以等在这然后出去：



另一方面，如果这个点（圆方树上的父亲）连向点双内所有点，那对于 Alice 在某个子树内的情况，Bob 一定可以一步走到连接更深子树的原点，然后就可以继续向下。从而如果 Alice 第一步不能找到一个好的点双或者上述结构（以 Bob 当前点为根考虑），则 Alice 必输。

Tom & Jerry(loj3406)

考虑 Alice 从这里出去后的情况。如果外面有个好的点双那直接赢了，否则又变成了刚才的情况，Alice 需要在换了根的情况下再找一个上述结构，形成如下形式。

但如果有这种情况，Alice 就能在这两个结构上来回绕，从而获胜。这样就完成了讨论。

Tom & Jerry(loj3406)

总结起来，我们需要判断点双是不是好的，需要对于点双的每个点判断 Bob 从这里进来时 Alice 能不能出去，然后需要判断每个能出去的结构能否和外面的部分进行配对。对于询问，可以先在圆方树上定位子树，然后相当于询问子树内这个方向上是否存在能 Alice 获胜的结构（好的点双或者这个方向且存在一个配对的结构），

由于完善和整合细节也是比赛能力的一部分，这里的完整细节请读者作为一个练习。

如果倍增实现最后的定位子树，复杂度为 $O((n+q)\log n)$