弹性碰撞 physics

首先注意到一点，由于碰撞后电性会发生反转，所以带正电的只会向左走，带负电的只会向右走。

先考察一种特殊情况：左边全是负电荷，类型分别是 $f_{1\cdots k}$。最后一个是正电荷，类型为 g。最后一个正电荷会发生碰撞，然后向右走，类型变成 $\underline{g}$（下划线表示翻转）。前面的第 $2\cdots k$ 个负电荷会碰撞两次（向右一次，反向后向左再撞一次），方向和类型最终不改变。第一个负电荷在类型翻转后会被收集。

然后考察一般情况。我们可以认为是第一个正电荷向左走，其它正电荷不动；再是第二个正电荷向左走，其它正电荷不动。以此类推，显然按照这样的逻辑得到的结果是不变的。

假设所有问号都被确定了，那么设正电荷共有 k 个，显然只有前 k 个电荷会被收集。又根据之前的结论，如果初始状态是正电荷，收集时类型一定为 A，否则一定为 B。

我们依次考虑每个电荷是否会产生贡献。有两个要求，第一个是这个电荷为负或问号，第二个是整个序列中的负电荷数量要大于等于它的下标。预处理组合数后缀和就可以做到 $O(n)$。

轻涟 vaguelette

首先思考 $F(T,a)$ 怎么求。如果一个方案中两条路径 $a\to b,c\to d$ 相交了，肯定是不优的。把两条路径异或起来，可以得到 $a\to c,b\to d$ 的不相交路径，这样做更优。按照这样调整，最优方案中所有路径都是不交的。换而言之，一条边至多被经过一次。

这提示我们对每条边分别考虑。对于边 $x\to fa_x$，把它断掉后会形成两棵子树，设两棵子树在 a 中出现次数分别为 k 和 $m-k$。如果 k 为偶数，那两棵子树之间可以内部匹配，这条边就不会被经过。否则必然有一条路径经过它。

这样我们就得到了一条边产生贡献的条件：它的子树结点在 a 中的出现次数为奇数。

继续深入探讨。在 $G(T,a)$ 中，假设 $siz_x=s$，求 $x\to fa_x$ 的贡献次数。不妨枚举 a 中恰好有 i 个数属于 x 的子树，这样的方案数就是 $\binom{m}{i}s^i(n-s)^{m-i}$。先除去 $i=0$ 或 $i=m$ 的情况。然后考虑有多少子序列符合条件。对于 s 中的部分，它要选出奇数个数，根据对称性，答案是 2^{i-1}。对于 s 外的部分，要保证子序列长度为偶数，根据对称性答案是 2^{m-i-1}。综上，我们可以列出式子：

$$
\begin{aligned}
w(n,m,s)&=\sum_{i=1}^{m-1}\binom{m}{i}s^i(n-s)^{m-i}2^{i-1}2^{m-i-1}\\
&=2^{m-2}(n^m-s^m-(n-s)^m)
\end{aligned}
$$

我们惊讶地发现，一条边的贡献次数只跟它子树的大小和总共的结点个数有关。

分子树内和子树外两部分考虑。设 $f_{i,j}$ 表示以点 i 为根，仅考虑 i 的子树，连通块大小为 j 的方案数；$g_{i,j}$ 表示以点 i 为根，不考虑 i 的子树，连通块大小为 j 的方案数。枚举每条边 $i\to fa_i$，只需要将 f_i 和 g_i 拼起来就可以得到答案。朴素实现是 $O(n^3)$ 的。

使用任意模数 NTT 可以做到 $O(n^2\log n)$。

考虑优化。注意到 f_i,g_i 和它们的转移相当于多项式乘法，且它们的次数 $\le n$。考虑维护 $0\sim n$ 这 $n+1$ 个点值就可以得到这个多项式。同时，单次乘法就变成了点值对应相乘，变成了 $O(n)$。注意到总的乘法次数是 $O(n)$ 级别的，所以这一部分的时间复杂度为 $O(n^2)$。

问题在于我们还要去还原多项式再计算贡献，造成了时间复杂度瓶颈。这个问题也是好解决的，只需要把每个点的多项式乘上对应的边权后再加起来，最后统一还原，就可以做到 $O(n^2)$。

很好，现在解决了第一问。第二问也是类似的，考虑每个连通块的贡献。枚举需要算贡献的边 $x\to fa_x$，设 $f_{i,j,k}$ 表示考虑到点 i，上面的连通块大小为 j，下面的连通块大小为 k 的方案数。

对于 $i<x$ 的部分，它只能连向上面的连通块或者不在连通块内，方案数为 j 或 $i-1$。对于 x 时，需要连接上面的部分，方案数是 j。当 $i>x$ 时，可以选择不连，连上面或者连下面，方案数分别是 $i-1/j/k$。朴素实现是 $O(k^4)$ 的。

注意到我们的转移跟 x 并没有太大关系。考虑优化状态，设 $f_{i,j,k,0/1}$ 表示考虑到点 i，上面为 j，下面为 k，是否确定 x 的方案数。注意确定 x 时需要乘上 val_x。总时间复杂度 $O(k^3)$。

树上二维偏序问题 partial

首先考虑暴力怎么做。

可以发现一个结论：如果一个 `?` 填的是 0，那么它祖先的 `?` 也就是 0。证明是显然的。

记 $c_{i,0/1/2}$ 表示结点 i 的祖先中 0/1/? 的数量，$d_{i,0/1/2}$ 表示子树中的数量。（均不包含自身）先钦定所有问号都填 1，此时每个 0 的贡献为 $f_i = d_{i,1} + d_{i,2}$。从根往下依次考虑每个问号是否被修改。对于一个结点 i，考虑它从 1 变成 0 后的变化量，可以得到 $\Delta i = (d_{i,1} + d_{i,2}) - (c_{i,0} + c_{i,2})$。

容易发现一个结点，它的变化量一定不大于祖先的变化量。所以我们直接把所有 $\Delta i > 0$ 的问号结点全部取反就行了。答案为 $\sum_{a_i=0} f_i + \sum_{a_i=?} \max(\Delta i, 0)$。每次修改重新计算 f, Δ 的值可以做到 $O(nq)$。

接下来考虑链怎么做。

每次修改一个点后，可能会使得它的祖先 f 变化 1，祖先或子树的 Δ 变化 1。这不太好用数据结构维护，考虑询问分块。设立阈值 B，将询问涉及到的至多 B 个点作为关键点。可以发现关键点将链分成了至多 $B+1$ 段，每一段中的变化情况是一样的。

问题在于怎样快速处理块内 f, Δ 加减 1 的操作对答案的影响。修改 f 是简单的，只需要记录块中 $a_i = 0$ 的数量即可。修改 Δ 可以考虑对每一块预处理块内 Δ 整体 $+x$ 后对答案的贡献。具体地，维护一个桶 b_i 表示块内 $\Delta = i$ 的点的数量。那么整体 $+x$ 的贡献就是 $\sum_{i+x\geq 0} b_i(i+x)$。维护 b_i 和 ib_i 的后缀和可以得到答案。

注意到 x 是 $O(B)$ 级别的，所以只需要计算 $O(B)$ 个后缀和即可。

总时间复杂度 $O(\frac{nq}{B} + qB)$，取 $B = \sqrt{n}$ 可以做到 $O(q\sqrt{n})$。

最后考虑一般的情况。

把链上的思想拓展到树上，我们把这些关键点的虚树建出来，那么虚树上至多有 $2B$ 个结点。我们把每对相邻关键点路径上的非关键点形成的连通块压成一块；对每个关键点中，所有不含关键点的子树压成一块，至多有 $4B$ 个块。然后使用类似于链的做法就可以做到 $O(q\sqrt{n})$。