Zbox loves Keyboard

首先一个显然的结论就是假设没有 backspace 操作的话就是一个智障的递推，然后我们可以发现加上 backspace 之后的后效性并不严重所以我们考虑用类似 spfa 的方法优化转移，一个显然的结论是 backspace 只会用很少的次数所以直接对 n+100 以内的转移建边

考虑复杂度，总进队次数是 nlogn 次而建边的复杂度是调和级数 nlogn 所以总复杂度是 O(nlog^2n)

然后可以根据题目性质进行一些常数优化来通过本题

Zbox loves graph

直接考虑为什么这个暴力可以过

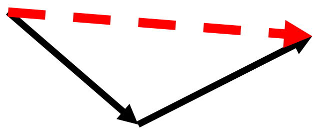
先 tarjan 把这个图变成 dag 然后对 dag 上的每个点开一个 bitset 表示这个点能到达哪些点然后按照逆拓扑序依次用 or 操作计算每个 bitset 的值，计算的时候把每个点的出边打乱，假设现在 v 是 u 的一个后继还没更新，然后 u 已经可以到达 v 了就不需要拿 u 的 bitset 来 or 上 v 的

然而还是不能过，所以不要直接做上面的步骤，要把每个弱联通分量中的点重标号以后分开做(意义就是 bitset 的大小可以更小)

首先如果点数多那么边就很稀疏每个弱连通分量都很小所以 bitset

的大小很小就可以过

如果点数不多图就很稠密类似下图的边就有很多，就有大量的边不会导致一次 bitset 的 or 操作然后就可以过



Zbox loves memory

先考虑所有的 0 操作都在 1 操作之前的情况，我们显然可以用~~二进制分组~~线段树套 trie 在 O(Tlog^2T)内完成

我们考虑一个对时间分块的做法，假设块大小为 P，我们维护一个上述结构和一个记录 insert 操作的数组

一个时刻如果遇到一个 insert 就加到数组里，否则就在上述结构中查询答案然后在数组中暴力查询答案打擂输出

如果当前时刻是 P 的倍数我们就清空这个数组然后对所有 insert 操作重建上述结构

重建上述结构可以利用 trie 的合并做到 O(sizelogsize)，写一个可持久化的 trie 即可

考虑复杂度

单个操作的复杂度是 O(log^2T+P)

重建结构的总复杂度是 O(T/P*TlogT)

取 P 为 sqrt(TlogT)可以达到最优复杂度 O(Tsqrt(TlogT))

具体下标的问题可以用一个平衡树来维护不影响复杂度

块链套 Trie 也可以解决本题。

1.1 subtask1

暴力枚举每次删掉哪个数。

时间复杂度 $O(n!)$ 。

1.2 subtask2

枚举最终有哪些数留下来。对于被删去的数的顺序，除了不能让某些数最后一个被删掉外都可以任意排列。

时间复杂度 $O(n2^n)$ 。

1.3 subtask3

用 DP 优化 subtask2 。$f_{i,j,k}$ 表示前 i 个数中留下 j 个数且第 i 个被留下，被删去的数中恰有 k 个不能最后被删去的方案数。

转移简单，此略。

时间复杂度 $O(n^4)$ 。

1.4 subtask4

再优化 $subtask3$ 。若 A 的某个子序列 S 最终可能被留下来，则 S 相邻两数之间至多只有一个数不能在最后被删去。

先假设这类数可以在最后被删去。则用 $f_{i,j}$ 表示前 i 个数中留下了 j 个数且第 i 个被留下的方案数，答案很好统计。那么我们的任务即统计出有多少种情况是不合法的，然后减掉即可。

枚举不合法的方案中，最后删掉的元素是哪一个，然后枚举前缀选了多少个，后缀选了多少个，即可统计答案。

时间复杂度 $O(n^3)$ 。

1.5 subtask5

假设我们将 A 删成非降序列后依然可以继续操作。那么只需要用类似 subtask4 中 $f_{i,j}$ 的方程统计答案即可，用树状数组可以优化 DP 至 $O(n^2 \log n)$ 。

现在我们考虑怎么减去不合法的答案，即将 A 删成非降序列后依然删去了元素。再定义 g_k 表示留下 k 个数的合法答案有多少种，则不难用容斥得出 g_k 的递推式：

$$g_k = \sum_{i=1}^{n} f_{i,k} - \sum_{j=k+1}^{n} \binom{j}{k} \times g_j \times (j-k)!$$

总时间复杂度 $O(n^2 \log n)$ 。