A、换座位

SOURCE：瞎编的。

这题的线性做法是这样：

我们首先从前向后模拟，确定一下一直执行到每次询问的 t_i 时候，现在 id_i 跑到第几个位置上去了。

然后我们再从后向前模拟，确定一下执行完最后 $m-t_i$ 次操作后，每一个位置的人，现在去了哪个位置。

两个拼一下就是答案。

B、蛇形数组

SOURCE：瞎编的。

这题实简单。

我们考虑把每次操作对应的坐标抽象成数字 id（这个需要算算），那么，每次查询，实际上就是想知道，一个初始是全 0 的序列里面，第 id 个 0 在哪里，然后把那个 0 改成 1。

那么，直接用线段树维护即可，需要动态开点。

C、找不同

SOURCE：别人问我的。

前 30 分不说了。

先嘴一个也许能过的做法，但是我没写。

假设要求最近，则我们直接从所有字符串对应的二进制状态出发，一起`BFS`，找到距离每个点最近的点即可。

现在要求的是最远，等价于在翻转的二进制状态下要求最近，那岂不是一样的？只不过这次目标点的集合不一样了罢了。

正解的话是这样，我们定义一个 dp。

$dp_{i,sta}$ 表示考虑了跟 sta 二进制下前 i 位不一样，其他位一样的所有数字，跟 sta 差异度最大的状态有多大。

那么 $dp_{i,sta}$ 转移其实就两种，要么是 $dp_{i-1,sta}$，要么是 $dp_{i-1,sta\bigoplus 2^i}+1$。

初始状态是：$dp_{0,sta}=0$，如果 sta 在输入中存在，如果不存在则是 $-\inf$。

可以通过滚动压缩一维。时间复杂度$O(m2^m)$

D、切割字符串

SOURCE：ABC240EX

容易想到的一个暴力做法是：$dp_{i,j}$ 表示考虑到 i 了，一共选了 j 个区间，结尾字符串最小是谁，也就是保留一个字符串。

显然 j 不会超过 2650，这个可以暴力枚举算一下。

此外，对于相同的 i，随着 j 变大，$dp_{i,j}$ 必然字典序单调。

以及，$dp_{i,j}$ 对应的字符串，长度不会超过 j。

那么我们考虑在做这个暴力 dp 的时候优化一下转移：

考虑维护一个 dp 数组，存的是对于当前 i 来说，每一个 dp_j。

我们顺着转移：

对于每一个 i，我们暴力枚举 k 从 $i+1,\dots,n$，看看 $S_{[i+1,k]}$ 形成的字符串可以拼在哪一个 $dp_{i,j}$ 的后面。

显然，由于 dp 数组本来是单调的，所以插入位置有且只可能有一个，我们直接双指针找到对应的 k 即可。

一个细节是，我们是用顺推的形式转移，所以需要把每一次可以更新的内容，丢到一个 $lazy$ 性质的 $vector$ 里，当枚举到 i 的时候再更新这些内容到 dp 数组中。