合成

可以考虑从结果往前推。

以`j`为例，第一次变成`lb`或`bl`。

以`lb`为例，第二次变成`bjl`。

第三次变成`jlbj`。

第四次变成`lbjlb`。

第五次变成`bjlbjl`。

通过观察可以发现，序列a合法，当前仅当对于任意三个相邻元素，j,l,b各自出现且仅只出现一次。

异或

$n=1$时，无解。

n为偶数时，输出n个相同的数。

n为奇数时，输出$3,7,4$，然后输出$n-3$个相同的数。

作业

总体思想是，类比$1111=(10000-1)/9$，所以只需要在式子里构造一些$R-1$即可。

$$
\begin{aligned}
S=\sum_{i=0}^{n-1}\overline{nn\cdots n}(i+1\text{个}\,n)&=\sum_{i=0}^{n-1}\sum_{j=0}^{i}n\cdot R^j\\
&=\sum_{0\le j\le i\le n-1}n\cdot R^j=\sum_{j=0}^{n-1}\sum_{i=j}^{n-1}n\cdot R^j\\
&=\sum_{j=0}^{n-1}(n-j)n\cdot R^j=n\sum_{j=0}^{n-1}(n-j)R^j\\
&=n\sum_{j=0}^{n-1}R^jn-R^jj=n(\sum_{j=0}^{n-1}R^jn-\sum_{j=0}^{n-1}R^jj)\\
&=n(n\sum_{j=0}^{n-1}R^j-\sum_{j=0}^{n-1}R^jj)
\end{aligned}
$$

$$
\begin{aligned}
\sum_{j=0}^{n-1}R^j&=\frac{1}{R-1}\sum_{j=0}^{n-1}R^j(R-1)=\frac{1}{R-1}\sum_{j=0}^{n-1}(R^{j+1}-R^j)\\
&=\frac{1}{R-1}(\sum_{j=0}^{n-1}R^{j+1}-\sum_{j=0}^{n-1}R^j)=\frac{1}{R-1}(\sum_{j=1}^{n}R^j-\sum_{j=0}^{n-1}R^j)\\
&=\frac{1}{R-1}(R^n-1)
\end{aligned}
$$

$$
\begin{aligned}
\sum_{j=0}^{n-1} R^j j &= \frac{1}{R-1}\sum_{j=0}^{n-1} R^j j(R-1) = \frac{1}{R-1}\sum_{j=0}^{n-1} R^{j+1}j - R^j j \\
&= \frac{1}{R-1}(\sum_{j=0}^{n-1} R^{j+1}j - \sum_{j=0}^{n-1} R^j j) = \frac{1}{R-1}(\sum_{j=1}^{n} R^j(j-1) - \sum_{j=0}^{n-1} R^j j) \\
&= \frac{1}{R-1}(R^n(n-1) + \sum_{j=1}^{n-1} R^j(j-1) - R^j j) = \frac{1}{R-1}(R^n(n-1) - \sum_{j=1}^{n-1} R^j) \\
&= \frac{1}{R-1}(R^n(n-1) - (\sum_{j=0}^{n-1} R^j) + 1) \\
S &= n(n\sum_{j=0}^{n-1} R^j - \frac{1}{R-1}(R^n(n-1) - (\sum_{j=0}^{n-1} R^j) + 1)) \\
&= n((n + \frac{1}{R-1})\sum_{j=0}^{n-1} R^j - \frac{1}{R-1}(R^n(n-1)+1)) \\
&= n((n + \frac{1}{R-1})\frac{1}{R-1}(R^n-1) - \frac{1}{R-1}(R^n(n-1)+1)) \\
&= n\frac{R^{n+1} - (R-1)n - R}{(R-1)^2}
\end{aligned}
$$

对决

a. 满分做法

随机输出R, P, S。你将有高达$3^{-2\cdot 10^5}$的概率获得满分。

b. 简单做法

注意到扰动的概率实际上并不是很高，一个段也比较长。所以我们可以考虑**记忆一些连续的出手**。例如把连续五次出手称为组，把所有组都记录下来。在每次出手时，取前四次交互库的回应，然后尝试填入R, P, S，看看哪种组出现过的次数最多，并假设交互库下次会这样出。

这个做法还可以更加完善和科学。例如综合考虑连续$1-20$次出手，并**为不同长度的组加不同的权**。经过测试，$f(len) = 2^{len}$是一个不错的选择。这样的做法可以达到约80%的胜率，获得约56分。

c. 复杂做法

在这个做法中，我们考虑真正利用到题面的设定。我们可以通过交互库给出的已知信息，**尝试还原出原有的“段”**。注意，我们还原得到的段不一定是准确的。因此，随着信息的增多，我们可以多次尝试还原，使得结果逐渐逼近真实结果。还原的频次应当以卡在时限以内为准。

关于还原原有的“段”，一种可行的办法是，从头开始，先枚举第一段的长度，然后在整个已有的交互库回答中**模糊匹配**，找到一个长度，使得匹配次数最多，匹配度最高；然后以此类推。判定模糊匹配的一种可行办法是，求两个字符串的**编辑距离**，这个使用动态规划即可。注意：这里的求编辑距离和一般求法可以不同。因为如果是需要寻找的段，答案必然很小，因而可以采取不同的动态规划策略。判定模糊匹配的另一种可行办法是，求**最长公共子序列**长度。同理，由于本题中的字符串都是随机生成的，所以可以直接用指针扫描并做调整，得到一个线性的复杂度。

理论上来讲，通过b做法的变形也可以达到类似的效果，也即，把匹配改成模糊匹配。但是这样带来的问题是，所有的组不再能用map存储，时间复杂度要达到$O(n^2)$甚至$O(n^3)$，很拉胯。因此c做法的优势在于，利用题面保证段的数量不超过10个，求出原有的段以减少需要的比较次数。

通过这种做法，正确率应该可以达到90%或者更高。