NOIP 模拟赛

1　降水（rain）

根据题意，设答案为 $b_1, b_2, \cdots, b_n$ ，则可以列出方程:

$$a_1 = \frac{b_1}{2} + \frac{b_2}{2}$$
$$a_2 = \frac{b_2}{2} + \frac{b_2}{2}$$
$$\cdots$$
$$a_n = \frac{b_n}{2} + \frac{b_1}{2}$$

用 a_i 和 b_{i+1} 表示 b_i :

$$\begin{aligned} b_1 &= 2a_1 - b_2 \\ b_2 &= 2a_2 - b_3 \\ b_3 &= 2a_3 - b_4 \\ &\cdots \\ b_n &= 2a_n - b_1 \end{aligned}$$

消去 b_2 到 b_n :

$$b_1 = 2a_1 - 2a_2 + 2a_3 - 2a_4 + \ldots + 2a_n - b_1$$
$$b_1 = a_1 - a_2 + a_3 - a_4 + \ldots + a_n$$

所以我们可以用数据直接算出 b_1 ，然后在原方程组的基础上，用 a_i 和 b_i 表示 b_{i+1}：

$$
\begin{gathered}
b_2 = 2a_1 - b_1 \\
b_3 = 2a_2 - b_2 \\
b_4 = 2a_3 - b_3 \\
\cdots \\
b_n 2a_{n-1} - b_{n-1}
\end{gathered}
$$

将 b_1 代入，直接递推就可以算出所有答案，时间复杂度 $O(n)$ 。

2　查找（search）

令 x_k 为两个序列合并后第 k 小的数，a_i 为第一个序列的元素，b_i 为第二个序列的元素。

考虑找到最大的 i，使得 $\max(a_i, b_{k-i}) \leq \min(a_{i+1}, b_{k-i+1})$，则 $x_k = \max(a_i, b_{k-i})$。

发现 i 可以二分找到，复杂度 $O(n + q\log n)$。

当有多个序列时，还有 $O(k\log kn + q)$ 的原地归并做法和 $O(kn + qk\log n)$ 的多序列同时二分做法。

3　概率（pr）

3.1　sol1：组合数

我们只需要知道前 n 个和后 n 个相同的方案数，容易得到答案概率。

所有数在 $[0, m]$，前 n 个和后 n 个相同，可以看作前 n 个在 $[0, m]$ 中，后 n 个在 $[-m, 0]$ 中，和为 0 的方案数，给后 n 个元素加上 m ，等价于所有元素在 $[0, m]$ 中和为 nm 的方案数。

于是这就是经典的容斥，$0 \leq x_i \leq m$，$\sum x_i = nm$，钦定 k 个不满足上界限制，将他们减掉 $m+1$，方案数就是 $f(k) = \binom{2n}{k}\binom{2n+t-1}{t}, t = nm - k(m+1)$ ，答案就是 $\sum_{k=0}^{n}(-1)^k f(k)$。

3.2　sol2：生成函数

只用算出前 n 个数的和与后 n 个数的和相等的概率 p，那么答案即为 $(1-p)/2$。

记前 n 个数的和的生成函数为 $F(x) = \sum_{i=0}^{nm} f_i x^i = \left(\frac{1-x^{m+1}}{1-x}\right)^n$，那么 $p = \sum_{i=0}^{nm} f_i^2/(m+1)^{2n}$。看起来要把 $F(x)$ 的所有系数 f_i 都求出来，很困难。

但发现 $f_i = f_{nm-i}$，于是 $\sum_{i=0}^{nm} f_i^2 = \sum_{i=0}^{nm} f_i f_{nm-i} = [x^{nm}]F(x)^2 = [x^{nm}]\left(\frac{1-x^{m+1}}{1-x}\right)^{2n} = \sum_{i=0}^{2n}(-1)^i\binom{2n}{i}\binom{2n-1+mn-i(m+1)}{2n-1}$。于是预处理出 $O(nm)$ 范围内的阶乘即可 $O(n)$ 求出单组答案。

时间复杂度 $O(nm+Tn)$。

4 回文（pali）

4.1 20pts

暴力 $O(TL^7)$。

4.2 100pts

其他测试数据我乱给的，大概是给了一个 $O(TL^4)$ 一个 $O(TL^3)$ 一个我也不知道咋做。

先将三个字符串接一起变成大串 S，不然下标太复杂。考虑大力 DP，从回文串的两端向里匹配，设 $f_{i,j}$ 表示左边匹配到 i，右边匹配到 j 的方案数。大概的思路是选择要不要跳到下一个串去继续匹配。具体的转移有三种：

1. 不跳到下一个串，转移到 $f_{i+1,j-1}$，前提是都还在串内。

2. 一端跳到下一个串，以左端点为例，转移到 $f_{k,j-1}$，其中 k 大于 i 且不与 i 在同一个串且 $S_k = S_{j-1}$，可以开个辅助数组（大致是存一下在哪个整串的哪个字符，同时另一端在哪的方案数的和）优化一下。

3. 两端同时跳到下一个串（只能从 i 在 A，j 在 C 的情况跳到都在 B 的情况），从 $f_{i,j}$ 转移到 $f_{k,l}$，其中 k 和 l 都在串 B 上而且 $S_k = S_l$，开个变量就可以优化掉了。

最后统计答案的时候就是 $f_{i,i}$ 和 $f_{i,i+1}$（要求 i 和 $i+1$ 在同一个串）和 $f_{i,j}$（要求 i 和 j 在相邻的串）的和。

复杂度 $O(TL^2)$，需要注意实现细节。三次方和四次方做法大概就是没有用优化的复杂度，但是不比平方还好写吧，大概。