1． 不难发现， 如果我们知道正反面分别和几个庄家下了注，我们下的注一定是收益最多的几个庄家， 所以我们可以把两组庄家按照收益从大往小排序， 那么每一边的收益一定是一个前缀和。 这样我们可以通过枚举左右各取了几个， O(N²)的计算答案。

考虑加速这个过程，假设较小的收益是第一组庄家的前缀，我们可以从小往大枚举第一组取的个数，然后在右边找到要超过这个收益，最少要和几个庄家下注，这个可以用一个指针来扫。 这样我们可以在排完序之后线性时间内做完此题， 复杂度O(NlogN)， 瓶颈在于排序， 如果采用桶排序可以做到O(N)

2. 先去掉所有 s[i]=t[i]的文件。如果有一个文件 i 的目标 t[i]等于某文件 j 的 s[j]，就连边 i->j。形成的图中一定是若干链或环，环需要额外一次操作。

3.

正解，区间dp，将问题转换成每次添入一个数，状态 $f[l][r]$ 表示在 $l-1, r+1$ 位置上的数已经放置的前提下区间 $[l, r]$ 内的方案数，预处理出来组合数,定义$c_n^m = c[n][m]$，状态转移就是枚举出来一个数 k，是在该区间内选取的第一个数

$$f[l][r] = \sum_{k=l}^{r} f[l][k-1] * f[k+1][r] * c[r-l][k-l]$$

因为 k 位置的数是直接确定好的，所以不参与计算，区间内的元素除 k 位置外一共有 $r-l$ 个，对于每个确定的方案a和方案b，方案内部顺序不变，然后两个方案互相交错，例如：方案一的添加顺序是[1,2,3] 方案二的添加顺序是[4,5,6,7],则一种总顺序可能为[1, 45 ,2,3 67]。

则方案数就是 C_{l-r}^{k-l} 就是在 $l-r$ 个填入位置中，使 $k-l$ 个数选中 $k-l$ 个位置，并在这些位置中，按添加顺序拜访，上文给的例子可理解为选了1,4,5这三个位置，然后将方案一的顺序[1,2,3]填入。

以及要处理边界问题，提前将 $f[i+1][i]$ 赋值为1（原因根据状态转移方程自行理解一下）

以及，要提前去掉原序列中，相邻且相同的巧克力块，并且判断该盒巧克力答案是否为0（详情见标称）

4. 对每个位置求以该位置为右下角和左上角的矩形个数。以右下角为例。
每次求解一整行的答案。设当前求i行的答案。从左往右，枚举到(i,j)时，维护从位置(i,j)往上的竖线$(k,j)|1 \le k \le i$，竖线上每个位置(k,j)维护以(k,j)为右上角，以(i,j)为右下角的“C形”数量。竖线右移一位，需要进行区间加、单点改、区间求和，使用线段树实现。时间复杂度$n^2 logn$