1001. Maximum Multiple

$1=\frac{1}{2}+\frac{1}{3}+\frac{1}{6}=\frac{1}{3}+\frac{1}{3}+\frac{1}{3}=\frac{1}{2}+\frac{1}{4}+\frac{1}{4}$.

1002. Balanced Sequence

对于一个括号序列，令n是长度，m是前缀最小值('(' +1，')'-1)，sum是最后的和，那么答案就是$m-sum+2m$。现在这个n和sum都是定值，只要最大化m就好了。 每个串，把匹配后的搞掉之后，会得到一个pair (a, b)，表示a个左括号接上b个右括号，把(a, b)按照一定顺序排一排依次接起来就知道m的最大值了。

1003. Triangle Partition

求个凸包，然后选择凸包一条边AB，然后找个和AB夹角最小的点C，把ABC当做一个三角形删掉即可。这样做个n次，显然这样求出来的三角形们是合法的。

1004. Distinct Values

显然可以从左往右贪心，问题转化成求一个区间里的mex，由于区间左右端点都是递增的，用个set维护下即可。

1005. Maximum Weighted Matching

一个显而易见的观察是：按照操作来可以很容易进行dp。于是只要还原出操作序列即可：每次选个度数为2的点删掉，然后加入一条虚边。 $dp(edge, x, y)$表示处理到$edge$这条边，这条边左端点匹配状态是x，右端点匹配状态是y的最大匹配和方案数。碰到重边的时候merge下即可。

1006. Period Sequence

考虑这个无限长的序列中相等的数$s_{i_1}, s_{i_2}, \ldots, s_{i_k}$，考虑$i_1 \bmod n, i_2 \bmod n, \ldots, i_k \bmod n$，这些构成了一条链。相邻两个位置有个$delta$，表示下一个相同位置在$delta$后。

然后考虑这些链，肯定是$s_i \bmod n$一样的数组成的。不妨$O(n^2)$枚举这个链的一部分，统计它的贡献。设距离上一个位置是$prev$，距离下一个位置是$next$，这时候会分成4种情况：

- 这个prev and next都在$[a, b]$内：贡献是一个等差数列乘以若干常数
- prev部分在$[a, b]$里，next全在$[a, b]$内：贡献是两个等差数列相乘，然后乘以若干常数
- prev完全在$[a, b]$内，next部分在$[a, b]$内：贡献是两个等差数列相乘，然后乘以若干常数
- prev和next都部分在$[a, b]$内：贡献是三个等差数列相乘，然后乘以若干常数

具体等差数列就不列出来了，有std的参考std，没有的找有std的人要。

1007. Chiaki Sequence Revisited

考虑这个数列的差分数列，除了个别项，本质就是：$1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, \ldots$。

可以观测到，这个序列可以这么生成：一开始只有一个1，1变成110，0保持不变。迭代无穷多次后就是这个差分序列。

知道差分序列，可以应用阿贝尔变换，把a的前缀和搞成差分序列相关。不妨令差分序列是da，那么a的前缀和$s(n) = (n-1)\sum_{i=0}^{n-2} da(i) - \sum_{i=0}^{n-2} da(i)i + 1$。

利用da的分形结构，很容易算出$s(n)$。

1008. RMQ Similar Sequence

RMQ-Similar实际上就是A和B的笛卡尔树一样，这样我们就有了一个二叉树，然后可以在树上分析了。

考虑到B中有元素相同的概率是0，于是可以假设B里面元素互不相同，也就是说可以假定是一个排列。

显然，符合笛卡尔树的排列就是这个树的拓扑序列个数，就是$\frac{n!}{\prod size_i}$。然后显然每个排列期望的和是$\frac{n}{2}$，于是答案就是$\frac{n}{2\prod size_i}$。

1009. Lyndon Substring

这边有个简单的结论，考虑$s = w_1 w_2 \ldots w_n$是s的lyndon factorization，那么$|w_i|$的最大值就是s的最长lyndon substring。证明略，想知道的私信我。

于是你只要会合并两个lyndon factorization就好了。这个是个经典问题，利用lyndon factorization的单调性，只要两个二分就搞定了。

1010. Turn Off The Light

考虑只有一个起点的时候应该怎么做。令起点在p，最左边的1在位置s，最右边的1在t，显然$p \to s \to t$或者$p \to t \to s$都是可以的，不妨只考虑$p \to s \to t$。

显然，先走到s，中间该取反的就取反，然后从s走到t，该取反的也取反，这些是要走的必要步数。考虑剩下来那些位置中1所在的位置为$i_1, i_2, \ldots, i_m$.我们要把这些位置的状态搞对，那么考虑相邻两两配对，来回一趟就可以把这两个都搞定。如果位置是奇数个，最后还需要和t搞一下。可以证明这样是最优的。

这个过程很容易就可以推广到起点不定的情况，考虑起点p和s和t的位置关系。

- 如果$p \le s$，那么p和s之间每个位置在过完$p \to s \to t$后都是1，剩下部分是个定值，可以直接维护。
- 如果$s \le p \le t$，显然在p从s慢慢增加到t的时候，1位置的变换也是很好维护的
- 如果$p > t$，这个时候显然不如走$p \to t \to s$逻辑，所以可以在另一种情况下做。

1011. Time Zone

转换成分钟，然后随便算算就好了。