1001 X-liked Counting

直接计算前缀或后缀中至少有一个满足条件的总数不太方便，考虑容斥，先计算没有任何一个前缀和后缀满足条件的数的个数，最后用总情况数减去它。

由于前缀与后缀间有重合的部分，所以不能直接在头尾两端同时转移。不妨先枚举整个数模 x 的结果，再从前或从后任选一个方向转移，并记录当前前缀或后缀对 x 的取模结果，这样前缀和后缀模 x 的结果都可以表示出来。转移时注意一下不能带前导 0 的细节。

总的状态数可能有些多，但是注意到许多状态是无用的，做些小的剪枝就可以顺利通过了。

1002 Buying Snacks

设$dp_{i,j}$表示当前考虑完前i个零食，一共用了j 块钱的方案数，可以得出复杂度为$O(n^2)$的转移方程：

$dp_{i,j} = dp_{i-1,j} + dp_{i-1,j-1} + dp_{i-1,j-2} + dp_{i-2,j-1} + 2dp_{i-2,j-2} + dp_{i-2,j-3}$

考虑如何去优化这个转移。

设F_i为dp_i的生成函数

下面有两个做法：

方法一：可以得到生成函数的转移方程：

$F_i = (x^2 + x + 1)F_{i-1} + (x^3 + 2x^2 + x)F_{i-2}$

对该递推式使用矩阵快速幂优化转移即可

这样做可能常数会比较大，有一些优化常数的方法：

发现复杂度主要来源ntt的计算复杂度，考虑如何减少使用ntt的次数。可以发现，在进行矩阵运算时，对于两个2×2的矩阵，我们一共需要进行8×2次dft和8次$idft$一共24次ntt运算，而dft其中有一半都是重复计算，我们可以先预处理出两个矩阵的dft，这样可以优化掉8次ntt运算。我们还可以发现，矩阵运算中只用到了乘法和加法，而乘法和加法对于点值表示是一样可以计算的，所以我们可以在一个矩阵的一个值全部加完以后再进行$idft$运算，这样可以节省4次ntt运算。所以常数就被优化了一半。

同理，可以发现快速幂的时候有一半运算是在计算一个矩阵的平方，对于这种情况则只需要进行4次dft运算。

方法二：可以利用倍增思想

对于F_{i+j}，我们考虑从中间断开。如果中间两个没有捆绑，则这时方案数的生成函数为$F_i * F_j$

如果中间两个捆绑为一体，则与之前的转移类似，此时生成函数为$(x^3 + 2x^2 + x)F_{i-1} * F_{j-1}$

所以我们可以倍增转移，每次可以从F_{i-2}, F_{i-1}, F_i转移到$F_{2i-2}, F_{2i-1}, F_{2i}$

这样也只需要算$O(logn)$次

两种做法最终复杂度都是$O(mlogmlogn)$

1003 Ink on Paper

一道简单的最小生成树题，题面可以转化成在一个完全图上求最小生成树，直接使用Prim解决，复杂度$o(n^2)$。

常数较好的kruskal也可以通过，但是如果是二分答案然后搜索做，复杂度o($n^2 log$答案)的可能过不了。

1004 Counting Stars

题目要求支持三种操作：1、区间求和 2、区间减$lowbit(a_i)$ 3、区间加2^k（$2^k \le a_i < 2^{k+1}$）

看到区间问题很容易想到使用线段树维护，但后面这两个操作都不是线段树支持的基础操作。

假设只有前两个操作，考虑如何维护。这也是一个较为经典的问题。对于像这种数值快速递降至稳定的函数（再例如区间开根号，区间求欧拉函数之类的），我们可以考虑进行暴力修改。分析一下复杂度：首先理解$x - lowbit(x)$是相当于将x二进制中的最后一位1删掉。可以发现，每个数最多被操作$logn$次就会变成0。因此一共操作次数只有$nlogn$次，加上线段树复杂度，我们可以$O(nlog^2n)$修改，$O(nlogn)$求和。实现方面，为了在暴力修改时忽略已经变成0的区间，我们可以在线段树上维护一个tag表示该区间是否所有数为0, 这个可以通过线段树标记下传维护。

那么考虑如何把第三个操作加进去。有了前面的思路，我们依然可以用二进制去考虑第三种操作。第三种操作相当于是将a_i最左边的1左移一格。容易发现第三种操作仅与a_i的最高位有关，与后面的位无关。所以容易想到把a_i的最高位和其他位置分开维护。那么第三种操作就相当于区间对最高位$\times 2$, 这个也是可以使用线段树维护的。那么这题就做完了，应该是一道思维难度不算高的数据结构题。

1005 Separated Number

总的答案不太好算，考虑算每一位对答案产生的贡献。由于每个数作为被分割的那一段的第几位产生的贡献都不一样，假设现在考虑的是整个数字的第 i 位 $d(0 \le d \le 9)$，它在所处的分割段的位权是 10^j，那么这时它的贡献就为 $d \cdot 10^j \cdot num$，其中 num 指代出现这种分法的总情况数。下面考虑怎么计算 num：

由于此时整个数字的第i位所贡献的位权为 10^j，所以可以知道第 $i+j$ 位后是一个分割点。假设 n 为输入数字的总长度，分两种情况：① $i+j<n$, 这时除了这个分割点我们还需要插入最多 $k-2$ 个分割点，所以 $num = \sum_{m=0}^{k-2} C_{n-j-2}^{m}$；② $i+j=n$，这时我们还需要插入最多 $k-1$ 个分割点，所以 $num = \sum_{m=0}^{k-1} C_{n-j-1}^{m}$。记 $F(a,b) = \sum_{m=0}^{b} C_a^m$，可以发现 $F(a,b) = 2 \cdot F(a-1,b) - C_{a-1}^{b}$，所以可以先预处理出所有的 $F(a,k-1)$ 和 $F(a,k-2)$，这样 num 就可以快速计算出来了。

对于 $i+j=n$ 的情况，直接将其计入答案；对于 $i+j<n$ 的情况，注意到此时 $10^j \cdot num$ 只与 j 有关，所以可以使用前缀和优化快速算出结果。

1006 GCD Game

换皮**Nim**，把每个数看成一堆石子，石子个数就是每个数的质因子个数。

提前用线性筛预处理一下10^7以内的质因子个数就行了。

1007 A Simple Problem

用线段树维护区间加，区间max。
考虑统计在两段之间的答案，前一段的后缀max和后一段的前缀max都是单调的。
可以在序列上二分，每次选择一段正好为k的区间，如果前一段的max更大，最优的区间不会在这段区间的左边，否则不会在这段区间的右边。
直接用这种方法合并答案，即可得到$O(qlog^3n)$的做法
如果能在线段树上二分，就可以做到$O(qlog^2n)$。
正解：
对于所有的询问，k是定值，将序列按k分段，超过n的部分补至k的倍数。
连续的k个数要么在一段内，要么在两段之间。
对于每一段建一棵线段树，每一段的线段树形态完全一样，二分可以在线段树上做，单次二分的复杂度为$O(logn)$。
每次修改只需要对端点处的$O(1)$段区间重新计算答案。
另外建一棵线段树维护n/k段的答案，需要支持区间加，单点修改，区间查询min。

对于修改，中间段做区间加，两端重新计算。
对于询问，中间段查询min，两端不足一个段的部分用二分计算答案
总复杂度$O(qlogn)$

1008 Square Card

可以发现要使正方形卡片有可能被一个圆完全包含，它的中心轨迹一定是一个圆，所以本题就转化为了计算两圆相交面积与第一个圆面积的比值。但其中还有一些细节，比如数据中奖励区域不一定能完全包含卡片，以及两圆位置的不同情况要分类讨论，具体细节可以看std。

1009 Singing Superstar

一道简单的字符串题，对母串中所有长度小于等于30的串做字符串哈希，对相同哈希值的串暴力统计答案，每个询问直接查询对应哈希值的答案。

也可以使用Trie/AC自动机来通过本题，数据范围放的很松，理论复杂度稍劣也能通过。

1010 Yinyang

因为黑色和白色都要连通，所以构成环的只可能是网格的最外面一圈。
由于$nm \le 100$，$min(n,m) \le 10$。
不妨设$n \ge m$，考虑DP。
DP时需要记录前m个格子的颜色和连通性，这样的状态不会超过20000个。
转移时枚举下一位置的颜色，不允许形成环，特判最后两个格子。