IOI2016 中国国家集训队第一次作业
试题泛做表格

长沙市雅礼中学 袁宇韬

2016 年 1 月 22 日

目录

1 非 Challenge 试题

试题编号	CLONES	试题名称	Attack of the Clones
题目大意	考虑所有 n 元布尔函数。令全集为所有 n 元布尔函数的集合。 令 Z 为所有满足 $f(0,\dots,0)=0$ 的函数的集合。 令 P 为所有满足 $f(1,\dots,1)=1$ 的函数的集合。 令 D 为所有对任意 $x_1,\dots,x_n$ 满足 $!f(x_1,\dots,x_n) = f(!x_1,\dots,!x_n)$ 的函数的集合。 令 A 为所有对任意 $i,a_1,\dots,a_n,b_1,\dots,b_n,c,d$ 满足如果 $f(a_1,\dots,c,\dots,a_n) = f(a_1,\dots,d,\dots,a_n)$ 则 $f(b_1,\dots,c,\dots,b_n) = f(b_1,\dots,d,\dots,b_n)$ 的函数的集合，其中 c 和 d 均为第 i 个参数。 给定一个包含 Z、P、D、A 和求交集、并集、差集、补集运算的表达式，求出表达式的结果中包含的元素个数。		
算法讨论	考虑题中给出的四个集合每个是否包含一个给定元素，则共有 $2^4=16$ 种情况。可以对这 16 种情况分别求出是否在所求答案中，再求出每一种情况对应的元素个数。 由于元素不在给定集合中的条件不好处理，可以只考虑元素在集合中的条件，再用容斥原理计算。这 16 种情况对应的数量都不难求出。 Z 集合与 P 集合中元素数量相同，均为 2^{2^n-1}，因为除了一个函数值固定外剩下的函数值可以任意选择。 D 集合中元素数量为 $2^{2^{n-1}}$，因为可以自由选择一半的函数值。 A 集合中元素数量为 2^{n+1}，因为所有 A 集合中的函数可以表示为 $(c_0+c_1a_1+\cdots+c_na_n) \bmod 2$，其中 $c_0,\dots,c_n$ 为 0 或 1。 剩余情况也可以类似求出。 求出每种情况是否在答案中可以将输入的表达式求值。实现时可以用一个二进制位表示一种情况。		
时空复杂度	时间复杂度 $O(nl)$，空间复杂度 $O(nl)$，其中 l 为表达式的长度。		

试题编号	MINESREV	试题名称	Minesweeper Reversed
题目大意	给定一个棋盘，其中有一些位置有地雷。 对于每个格子，如果这个格子或周围的八个格子中有地雷，则定义这个格子的覆盖集合为它本身，否则定义这个格子的覆盖集合为所有和它在同一个只考虑本身和周围都没有地雷的格子的四连通块中的格子以及它们周围的八个格子组成的集合。 对于每个格子，定义这个格子的被覆盖集合为所有覆盖集合包含它的格子的集合。要求选出最少的格子，使得这些格子的被覆盖集合的并集包含了所有的格子。		
算法讨论	显然对于每个有地雷的格子必须选择它本身。考虑所有本身和周围都没有地雷的格子形成的连通块，则选择一个连通块中的任何一个格子就可以覆盖整个连通块。对于本身没有地雷但周围有地雷的格子，选择这个格子可以覆盖它本身和与它相邻的所有连通块。 注意到每个这样的格子最多和两个连通块相邻，可以将连通块看作点，这样的格子看作边，则选择一条边可以覆盖两个点。注意到如果这条边的两个端点中有一个已经被覆盖，则选择这条边不会减少答案，可以要求所有选择的边没有公共点。这样选择一条边可以使答案减少1。 这是一个任意图的最大匹配问题，可以用带花树算法解决。		
时空复杂度	时间复杂度 $O(r^3c^3)$，空间复杂度 $O(rc)$，常数很小。		

试题编号	BB	试题名称	Billboards
题目大意	称一个长度为 n 的 01 串为合法的，如果任意一个长度为 m 的子串中至少有 k 个 1。求出有多少个合法的串满足 1 的个数最少。		
算法讨论	显然一种 1 的个数最少的合法方案为贪心地让每 m 个位置中的后 k 个位置为 1。令这 k 个 1 为一块。注意到所有合法方案可以看作这个方案中的每个 1 向左移动一些位置。在同一块中的 1 向左移动的距离一定递减。此外在连续两块中对应位置的移动距离一定递增，否则这两个 1 中间会有长度为 m 的一段只有 $k-1$ 个 1。 分两种情况讨论。如果 $n \mod m \le m-k$，则所有 1 最多向左移动 $m-k-(n \mod m)$ 的距离，否则最后 m 个位置中 1 的个数会不足。如果 $n \mod m > m-k$，则每一块中最后的 $m-(n \mod m)$ 个 1 不能移动，剩下的 1 可以最多移动 $m-k$ 的距离。 现在问题变为求出在 $p \times q$ 的矩阵中填入 0 到 s 的数，使得每一行、每一列均非严格递增的方案数。由钩子公式可以得到答案为 $$\prod_{i=1}^{p}\prod_{j=1}^{q}\frac{s+i+j-1}{i+j-1}$$ 注意到 p 很大而 s 很小，可以将大部分项消去，在 $O(qs)$ 时间内求出答案。		
时空复杂度	时间复杂度 $O(m^2)$，空间复杂度 $O(m^2)$。		

试题编号	YALOP	试题名称	Trial of Doom
题目大意	给定一个 $n \times m$ 的棋盘，每个格子是红色或蓝色的。你从左上角出发，每次可以走到一个八连通的格子。经过一个格子时会将这个格子和上下左右的格子反色。问你是否可以到达右下角并使所有格子变为蓝色。红色格子的数量很少。		
算法讨论	不妨假设 $n \leq m$。注意到 $n \geq 2$ 时一定可以通过八连通的路径走到任意格子而不影响路径上的其它格子。$n = 1$ 时还需要考虑解的奇偶性。下面只考虑 $n \geq 2$ 的情况。 如果前 i 列的格子是否选择已经求出，则可以通过第 i 列的格子颜色推出第 $i+1$ 列的答案。这样可以用第一列的答案表示出所有格子的答案，再解一个方程组就可以得到答案。由于红色格子的数量很少，可以求出每个红色格子对最后方程组的影响，再将这些影响合并。 注意到一个格子对最后方程组的影响关于这个格子离右边的距离有周期性。对所有可能的 n 打表可以发现这个周期很小 (不超过 16380)。可以预处理出一个周期内的影响，再对于每个红色格子得到对方程组的影响。 当 $n = 1$ 时还需要考虑解的奇偶性。需要求出每个红色格子对解的奇偶性的影响，以及有多组解时是否能得到满足条件的解。		
时空复杂度	时间复杂度 $O(k + n^2 + f(n))$，空间复杂度 $O(nf(n))$，其中 $f(n)$ 为预处理的周期。		

试题编号	SHORTCIR	试题名称	Shortest Circuit Evaluation
题目大意	给定一个包含一些变量的逻辑表达式，保证每个变量只出现一次且表达式为 CNF 或 DNF 形式。每个变量有一定概率为真，且相互独立。你需要按照一定顺序求这些变量的值，如果当前已经确定的变量可以确定某个子表达式的值，则不需要求出这个子表达式中剩余变量的值。 求出求值的变量个数的期望值的最小值。		
算法讨论	如果表达式只包含一种运算，则显然可以按照值为真的概率排序后按顺序求值。 当表达式有两层时，不妨假设表达式为 DNF 形式，即第一层的运算为 OR。一种贪心策略为在第二层的每个子表达式用上面的方法求出最优方案，再按照子表达式的 P/c 从大到小排序，其中 P 为这个子表达式为真的概率，c 为求出这个子表达式的值的期望步数。 注意到按照这种方案求出一个值后，剩下的问题是一个规模更小的子问题，且这个子表达式的 P 增大，c 减小，得到的排序结果不变。可以用数学归纳法证明这种方案的最优性。 这种方案只能在表达式最多两层时得到最优解，因为当表达式有更多层时不能保证求出一个值后排序结果不变。		
时空复杂度	时间复杂度 $O(l + n \log n)$，空间复杂度 $O(l + n)$，其中 n 为变量个数，l 为表达式的长度。		

<table>
<tr><td>试题编号</td><td>DIVISORS</td><td>试题名称</td><td>Something About Divisors</td></tr>
<tr><td>题目大意</td><td colspan="3">给定正整数 B 和 X，求出有多少正整数 N 使得 NX 至少有一个因子 D 满足 $N < D \le B$。</td></tr>
<tr><td>算法讨论</td><td colspan="3">令 $P = \gcd(D, X)$，$Q = \frac{D}{P}$，则由 $PQ \mid NX$ 可以得到 $Q \mid N$。令 $N = kQ$，则由 $N < D$ 得到 $k < P$，由 $D \le B$ 得到 $Q \le \frac{B}{P}$。因此 N 满足要求的条件为存在 k，P，Q 满足 $N = kQ$ ，$P \mid X$，$k < P$，$Q \le \frac{B}{P}$。显然应该选择尽量小的 P，令 $next(k)$ 为比 k 大的最小的 X 的因子，则应该选择 $P = next(k)$。
因此所有满足条件的 N 的集合为对于所有正整数 $k < P$，不超过 $\lfloor \frac{B}{next(k)} \rfloor$ 的正整数的 k 倍的集合的并集。可以先按照 $\frac{k}{next(k)}$ 分类，再用容斥原理求出 N 的个数。
直接实现这个算法会太慢，需要一些优化。注意到所有条件都是在一定范围内选择所有 k 的倍数，可以按照范围从大到小加入，这样只需要维护一次容斥的结果。进行容斥时，维护一些 (x, c) 表示所有 x 的倍数需要被选择 c 次，其中 c 可能为负。每次加入一个条件 k 表示选择所有 k 的倍数时，对于当前所有的 (x, c)，加入 $(\mathrm{lcm}(x, k), -c)$，表示需要减去同时满足的情况，再将所有 x 相同的项合并。这里可以进行一些优化，例如当 $\mathrm{lcm}(x, k) \ge B$ 时可以不加入，可以预处理出 X 范围内的 gcd 结果然后 $O(1)$ 计算 lcm。
这个做法可以轻松通过题目范围内的数据。</td></tr>
<tr><td>时空复杂度</td><td colspan="3">时间复杂度 $O(Xf(X))$，空间复杂度 $O(f(X))$，其中 $f(X)$ 为容斥原理计算时得到的不同元素个数。</td></tr>
</table>

试题编号	SHORT	试题名称	Short
题目大意	给定 n 和 k,求出有多少对整数 (a,b) 满足 $n<a<k, n<b<k$,且 $ab-n$ 是 $(a-n)(b-n)$ 的倍数。		
算法讨论	令 $x=a-n$, $y=b-n$, $m=k-n$, 则 $0<x<m$, $0<y<m$,且 $(x+n)(y+n)-n$ 是 xy 的倍数。 当 $n=0$ 时答案显然为 $(m-1)^2$,下面讨论 $n>0$ 的情况。由 $xy \mid (x+n)(y+n)-n$ 得到 $xy \mid xn+yn+n^2-n$。由于当 $n>0$ 时 $xn+yn+n^2-n>0$,可以得到 $xn+yn+n^2-n \ge xy$,即 $(x-n)(y-n) \le 2n^2-n$。不妨设 $x \le y$,则 $x \le n+\sqrt{2n^2-n}$,即 x 只有 $O(n)$ 种选择。 固定一个 x, 令 $d=\gcd(x,n)$, $p=\frac{x}{d}$, 则由 $xy \mid xn+yn+n^2-n$ 得到 $p \mid x+y+n-1$,即 $p \mid y+n-1$。由 $dy \mid xn+yn+n^2-n$ 得到 $y \mid \frac{n(x+n-1)}{d}$,可以得到 y 的上界。设这个上界为 t,注意到 t 是 $O(\frac{n^2}{d})$ 的。 在 $y \le \sqrt{t}$ 的部分可以枚举 y 并判断是否合法。由于已知 $y \bmod p$ 的值,这一步枚举的时间复杂度为 $O(\frac{\sqrt{t}}{p})$,即 $O(\frac{n\sqrt{d}}{x})$。在 $y>\sqrt{t}$ 的部分可以枚举 $\frac{t}{y}$ 并判断是否合法。由于可以用 $y \bmod p$ 计算出 $\frac{t}{y} \bmod p$,这一步枚举的时间复杂度与上一步相同。 因此总的时间复杂度为 $\sum_{x=1}^{n+\lfloor\sqrt{2n^2-n}\rfloor} \frac{n\sqrt{\gcd(x,n)}}{x}$,可以证明为 $O(n\tau(n)\log n)$,其中 $\tau(n)$ 为 n 的因子个数。		
时空复杂度	时间复杂度 $O(n\tau(n)\log n)$,空间复杂度 $O(1)$。		

试题编号	CNTHEX	试题名称	Counting Hexagons
题目大意	给定 N,L,X,K,求出有多少种方案选出一个在 L 到 N 之间的数和五个 1 到 X 之间的非递减的数,使得后五个数之和大于第一个数,且最多 K 个数相同。保证 $L>K$,且 $N-L$ 很小。		
算法讨论	由于 $N-L$ 很小,可以枚举第一个数。可以用容斥原理处理最多 K 个数相同的条件。首先枚举 5 的所有无序划分,然后可以按照划分要求一些数相同。由于 5 的无序划分只有 7 种,可以预先计算出容斥时的系数方便实现。由于在给定划分时容易计算出等价的方案被计算的次数,可以先忽略非递减的条件,再除以每种方案被计算的次数。 之后需要求出选出 1 到 X 之间的数 $a_1,\dots,a_m$ 使得 $c_1a_1+\dots+c_ma_m>s$,其中 $c_1,\dots,c_m$ 为划分的系数。可以先计算不考虑 s 的方案数,再减去不满足条件的方案数。可以对于每个 a_i 枚举这个数是否超过 X 进行容斥,这样问题变为求 $c_1b_1+\dots+c_mb_m \le s$ 的方案数,其中 $b_1,\dots,b_m$ 为正整数。 这个问题可以转化为求 $b_0+c_1b_1+\dots+c_mb_m=s+1$,其中 $b_0,\dots,b_m$ 为正整数。这是一个经典问题,可以用生成函数解决,这里不再详细介绍。由于划分方案很少,可以对每种情况计算出公式进行计算。		
时空复杂度	时间复杂度 $O(N-L)$,空间复杂度 $O(1)$。		

试题编号	BAKE	试题名称	The Baking Business
题目大意	有一些插入操作和询问操作。每个插入操作为 `I product_id.size_id province.city_id.region_id M/F age units_sold`，表示在编号为 `product_id.size_id province.city_id.region_id M/F age` 的地方增加权值 `units_sold`。每个询问操作为 `Q product_id.size_id province.city_id.region_id M/F start_age-end_age`，表示询问编号为 `product_id.size_id province.city_id.region_id M/F`，`age` 在 `start_age` 和 `end_age` 之间的权值之和。每个操作中的编号可以部分省略，对于询问操作表示询问所有被省略部分取任意值时的答案之和，对于插入操作看作被省略的部分是一个只出现这一次的特殊值。你需要回答所有询问。		
算法讨论	由于编号的层数较少而每层的可能编号较多，可以在插入时对所有能覆盖这个插入操作的可能询问更新答案。对于 `age` 部分可以用树状数组等数据结构维护。		
时空复杂度	时间复杂度 $O(n \log age)$，空间复杂度 $O(age)$ 。		

试题编号	PARSIN	试题名称	Sine Partition Function
题目大意	给定整数 m，n 和实数 x，求出 $$f(m,n,x) = \sum_{k_1+k_2+\cdots+k_m=n} \sin(k_1 x)\sin(k_2 x)\cdots\sin(k_m x)$$		
算法讨论	考虑 k_1 的值。如果 $k_1 = 1$，则所有情况在 $\sin x f(m-1,n-1,x)$ 中考虑到。如果 $k_1 > 1$，则由于 $\sin k_1 x = 2\cos x \sin(k_1-1)x - \sin(k_1-2)x$，这一部分可以用 $2\cos x f(m,n-1,x) - f(m,n-2,x)$ 表示。于是 $f(m,n,x) = \sin x f(m-1,n-1,x) + 2\cos x f(m,n-1,x) - f(m,n-2,x)$。可以用矩阵乘法优化递推。		
时空复杂度	时间复杂度 $O(tm^3 \log n)$，空间复杂度 $O(m)$。		

<table>
<tr><td>试题编号</td><td>LUCKYDAY</td><td>试题名称</td><td>Lucky Days</td></tr>
<tr><td>题目大意</td><td colspan="3">定义序列 S，$S_1=A$，$S_2=B$，$S_i=(XS_{i-1}+YS_{i-2}+Z) \bmod P (i\geq 3)$。有若干询问，每次询问满足 $l\leq i\leq r$ 且 $S_i=C$ 的 i 的个数，其中 C 对所有询问不变。</td></tr>
<tr><td>算法讨论</td><td colspan="3">当 $X=Y=0$ 时容易解决。当 $Y=0$ 时周期最大为 P，可以预处理一个周期内的结果。下面考虑 $X>0, Y>0$ 的情况。
由于当 $X>0, Y>0$ 时转移关系可逆，在一个周期后一定回到初始状态。令 G 为这个递推关系的转移矩阵，v 为初始状态向量，则周期 t 满足 $G^tv=v$。可以用大步小步算法求出 t。
还需要求出在一个周期内所有 $S_i=C$ 的位置。在状态向量中有一个值为 S_i，因此可以枚举所有可能的 P 中状态向量，然后找出 t 满足 $G^tv=v'$。同样可以用大步小步算法解决。
由于需要求 $O(P)$ 个 t，如果大步小步算法中步长为 s，则时间复杂度为 $O(Ps+\frac{P^2}{s})$。取 $s=\sqrt{P}$ 得到时间复杂度为 $O(P^{1.5})$。
解决一个周期内的问题后，所有询问可以用二分查找快速解决。</td></tr>
<tr><td>时空复杂度</td><td colspan="3">时间复杂度 $O(P^{1.5})$，空间复杂度 $O(P^{1.5})$。</td></tr>
</table>

<table>
<tr><td>试题编号</td><td>DOMNOCUT</td><td>试题名称</td><td>Colored Domino Tilings and Cuts</td></tr>
<tr><td>题目大意</td><td colspan="3">你需要用多米诺骨牌覆盖 $n\times m$ 的棋盘。定义一个覆盖方案的割为一条水平或竖直的穿过棋盘但不穿过骨牌的直线。你还需要给骨牌染色，要求相邻骨牌不能同色。你需要求出使得割数量最少的方案，割数量相同时要求使用的颜色最少。</td></tr>
<tr><td>算法讨论</td><td colspan="3">如果已经得到 $n\times m$ 的没有割的方案，很容易得到 $n\times(m+2)$ 的没有割的方案。只要将最右边的骨牌向右移动两格，在中间空白处放上骨牌。可以构造出 5×6 和 6×8 的没有割的方案，因此对于大部分情况都可以得到没有割的方案。对于剩下的所有情况可以搜索得到最优方案。
考虑染色方案。显然当最多使用两种颜色时只有可能是骨牌依次平铺，只能在 $\min(n,m)=1$ 或 $\min(n,m)=2$ 且 $2\mid n, 2\mid m$ 时得到最优解。对于剩下的所有情况可以搜索得到只使用三种颜色的方案。</td></tr>
<tr><td>时空复杂度</td><td colspan="3">时间复杂度 $O(nm)$，空间复杂度 $O(nm)$。</td></tr>
</table>

试题编号	SHORT2	试题名称	Short II
题目大意	给定质数 p，求出满足 $p<a$，$p<b$ 且 ab 是 $(a-p)(b-p)$ 的倍数的 (a,b) 数量。		
算法讨论	令 $x=a-p$，$y=b-p$，则 $x>0$，$y>0$ 且 $xy \mid (x+p)(y+p)$，即 $xy \mid p(x+y+p)$。分为三种情况。 当 $p \mid x$，$p \mid y$ 时，得到 $p^2x'y' \mid p(px'+py'+p)$ 即 $x'y' \mid x'+y'+1$。可以发现只有 5 组解。 当 $p \mid x, p \nmid y$ 时, 得到 $px'y \mid p(px'+y+p)$ 即 $x'y \mid px'+y+p$。令 $px'+y+p=kx'y$，则 $(ky-p)x'=y+p$。令 $d=ky-p$ ，则 $d \mid y+p$。又由于 $ky=d+p$，则 $y \mid d+p$。因此 $dy \mid (d+p)(y+p)$，而显然 $p \nmid d$，$p \nmid y$，所以每种 $p \mid x$ ，$p \nmid y$ 的方案会对应一种 $p \nmid x$，$p \nmid y$ 的方案。当 $p \nmid x$，$p \mid y$ 时同理。 当 $p \nmid x$，$p \nmid y$ 时，由 $xy \mid p(x+y+p)$ 得到 $xy \mid x+y+p$。可以得到 $xy \le x+y+p$，即 $(x-1)(y-1) \le p+1$。不妨设 $x \le y$，则 $x \le 1+\sqrt{p+1}$。设 $a+b+p=kab$，则 $(ka-1)b=a+p$，得到 $b \mid a+p$。令 $d=ka-1$，则有 $\min(b,d) \le \sqrt{a+p}$，即 $\min(b,d) \le \sqrt{p+\sqrt{p+1}}$。 当 $b \le d$ 时，枚举所有可能的 b。由 $d \mid a+p$ 得到 $a \equiv -p \pmod d$。由于 $a<b \le d$，a 只可能为 $d-p$，可以 $O(1)$ 判断。 当 $b>d$ 时，枚举所有可能的 d。同样有 $a \equiv -p \pmod d$。由 $d=ka-1$ 得 $a \mid d+1$ 即 $a \le d+1$，因此 a 最多有两个可能的值，可以 $O(1)$ 判断。		
时空复杂度	时间复杂度 $O(\sqrt{p})$，空间复杂度 $O(1)$。		

试题编号	HYPER	试题名称	Hypertrees
题目大意	超图是图的一种扩展, 其中一条边可以连接多个点。定义超树为一个连通的超图，但是去掉任意一条边后不连通。给定 n，求出有 n 个点的每条边连接三个点的超树数量。		
算法讨论	定义双连通超图为没有割点的超图。可以先计算出双连通超树的数量，再按照双连通分量计算出超树的数量。 一个至少有两条边的双连通超树的每条边连接的点中，一定有恰好一个点只连接这条边。如果有三个点, 则这个超树只有这一条边。如果有两个, 则去掉第三个点后会删除这条边, 由超树的定义得到会分成两个连通块, 与双连通性矛盾。如果一个都没有, 则为了保证去掉任意一个点后仍然连通，这三个点在去掉这条边后仍然连通，与超树矛盾。因此对于每条边一定对应一个点只属于这条边。 去掉每条边对应的这个点后，剩下一个双连通图。因此一个 n 个点 m 条边的双连通图对应一个 $n+m$ 个点 m 条边的双连通超图。因此需要计算出点数与边数之和不超过 n 的双连通图的数量。可以搜索得到。 接下来可以搜索由双连通分量组合为超树的方案，再乘上对应大小的双连通超树的个数，得到答案。 由于运行时间可能较慢，可以预先打表计算。		
时空复杂度	时间复杂度 $O(1)$，空间复杂度 $O(n)$。		

试题编号	CARDSHUF	试题名称	Card Shuffle
题目大意	有一个序列，初始时为 1 到 n。有一些操作，每个操作为 `a b c`，表示将第 $a+1$ 个到第 $a+b$ 个数移动到第 c 个数之后并逆序。求最后的序列。		
算法讨论	用 Splay 等支持区间操作的平衡树维护序列。		
时空复杂度	时间复杂度 $O(m\log n+n)$，空间复杂度 $O(n)$。		

试题编号	MISINT2	试题名称	Misinterpretation 2
题目大意	对一个串进行变换，将所有偶数位置的字母按顺序移动到开头，所有奇数位置的字母按顺序移动到结尾。问有多少由小写字母组成的长度 n 满足 $l \le n \le r$ 的串，使得这个串在变化后保持不变。		
算法讨论	显然只需要求出这个置换中的环的个数。当 n 为奇数时最后一位一定保持不变，因此 $n = 2m$ 时环的个数一定比 $n = 2m + 1$ 少 1。只需要对奇数计算答案。 n 为奇数时这个置换可以看作 $p_x = 2x \bmod n$。由于当 $\gcd(x, n) = d$ 时环的大小为 $\mathrm{ord}_2(d)$，因此环的个数为 $$\sum_{d\mid n} \frac{\phi(d)}{\mathrm{ord}_2(d)}$$ 首先预处理 n 的质因数分解，然后 $\phi(d)$ 容易求出。需要求出 $\mathrm{ord}_2(d)$。由于当 p，q 互质时 $\mathrm{ord}_2(pq) = \mathrm{lcm}(\mathrm{ord}_2(p), \mathrm{ord}_2(q))$，只需要对所有质数幂 p^c 求出 $\mathrm{ord}_2(p^c)$ 的值。注意到 $\mathrm{ord}_2(p^c) = \mathrm{ord}_2(p^{c-1})$ 或 $\mathrm{ord}_2(p^c) = p\,\mathrm{ord}_2(p^{c-1})$，只需要对质数 p 求出 $\mathrm{ord}_2(p)$，然后用 $c \log p$ 的时间得到 $\mathrm{ord}_2(p^c)$。 要对质数 p 求 $\mathrm{ord}_2(p)$，可以枚举 $p - 1$ 的所有质因子，对每个质因子 q 不断尝试除以 q 后是否仍然满足条件。这样需要 $O(\log^2 p)$ 的时间。		
时空复杂度	时间复杂度 $O((R-L)\log^2 R + \sqrt{R}\log R)$，空间复杂度 $O((R-L+\sqrt{R})\log R)$。		

试题编号	FINDSEQ	试题名称	Find a Subsequence
题目大意	给定一个长度为 n 的序列和一个 1 到 5 的排列，找到从左到右的五个数使得它们的相对大小关系和排列中相同，或输出无解。		
算法讨论	首先枚举第二个数和第四个数，则剩下三个数的位置区间可以确定，且不会相交。第一个数和第五个数中至少有一个是剩下三个数中最大或最小的，可以贪心地让这个数取到满足条件的最大值或最小值。另一个数同样可以贪心取满足条件的最大或最小值。第三个数只需要判断范围内是否有解，可以用前缀和得到范围内满足条件的数的个数。如果有解再用 $O(n)$ 时间找到解。 对于第一个数和第五个数需要预处理出一个前缀或一个后缀中大于一个数的最小值或小于一个数的最大值。用递推可以在 $O(n^2)$ 时间内求出。对于第三个数需要预处理出一个前缀中小于一个数的数的个数。同样可以 $O(n^2)$ 时间内求出。		
时空复杂度	时间复杂度 $O(n^2)$，空间复杂度 $O(n^2)$。		

试题编号	FLYDIST	试题名称	Flight Distance
题目大意	给定一个带权无向图，你可以修改边的权值使得每条边都是连接的两个点之间的最短路。求边权修改量之和的最小值。答案用分数输出。		
算法讨论	将一条边权值的改变量分为两个数 d_{i+} 和 d_{i-}，表示边权增加了 $d_{i+}-d_{i-}$。要求 $d_{i+}\geq 0$，$d_{i-}\geq 0$，边权改变量为 $d_{i+}+d_{i-}$。 可以将这道题转化为一个线性规划问题。对每一对点用一个变量表示它们之间的最短路长度，则可以对每条边得到一个条件。用单纯形算法可以解决。 由于本题需要用分数输出，需要自己实现分数。在分数约分时，可以预处理小范围内的最大公约数加速约分的计算。		
时空复杂度	时间复杂度 $O(\binom{2nm+\frac{n^2}{2}}{2nm})$，空间复杂度 $O(n^2m^2)$。		

试题编号	EVILBOOK	试题名称	Evil Book
题目大意	你有 n 个对手，击败每个对手需要 c_i 的代价，可以得到 m_i 的收益。你需要按顺序选择一些对手击败，使得你总共得到 666 的收益。你还可以选择用 x 的收益使 c_i 和 m_i 都变为原来的三分之一。问你的最小代价。		
算法讨论	显然对于 $m_i\geq 666$ 的人，应该在保证 $m_i\geq 666$ 的情况下尽可能减少。注意到 $x\geq 10$，一个 $m_i=666$ 的人最多只会被减少三次，否则不能获得收益。这样所有的状态只会有 4^n 次，表示每个被击败的人是被减少几次后被击败的。 注意到实际上的状态数达不到这个上界。如果 $m_i<222$，则这个人最多只会被减少两次，而如果 $m_i\geq 222$，则这样的人中不减少的个数最多有两个，否则已经达到 666 的收益。这样最多只会有大约 $\frac{n^2 3^n}{18}$ 个状态。		
时空复杂度	时间复杂度 $O(n^3 3^n)$，空间复杂度 $O(n^2 3^n)$。		

试题编号	CIELQUAK	试题名称	Ciel and Earthquake
题目大意	有一个 $R \times C$ 的网格图，现在每条边有 p 的概率被删除，问左上角和右下角连通的概率。		
算法讨论	注意到 R 很小，可以按照列进行状态压缩 DP。对每一行记录与这一行连通的编号最小的行，或起点。容易得到 DP 转移。 由于 C 很大，不能直接这样做。可以发现当 C 足够大时答案很接近 0，可以直接输出 0。 但是当 $R = 8$ 时答案减小速度很慢，不能通过本题。注意到每一步的转移都相当于对这一列的状态乘上一个转移矩阵，而每一列的转移矩阵都相同。当 C 增大时相邻两个 C 的答案之比会收敛到一个常数。可以发现在答案减小速度很慢时收敛速度很快，同样可以得到答案。 只需要预处理 $C \le S$ 的答案，然后可以得到所有 C 的答案。S 可以取 50。		
时空复杂度	时间复杂度 $O(SRf(R))$，空间复杂度 $O(Rf(R))$。		

试题编号	TSUBSTR	试题名称	Substrings on a Tree
题目大意	给定一棵树, 树上的每个结点有一个字符。定义这棵树的子串为从一个点开始每次走到一个孩子结点，在任意一个点停止后经过的点上面的字符连接得到的串。首先求出这棵树本质不同的子串的个数, 再给出一些询问，每次询问按照给定的顺序排列所有字母时字典序第 k 小的本质不同的子串。		
算法讨论	注意到给出的树是一个 Trie，可以对这个 Trie 建出后缀自动机, 然后两种询问都容易回答。关于对 Trie 建后缀自动机可以参考去年的集训队论文。		
时空复杂度	时间复杂度 $O(n)$，空间复杂度 $O(n)$。		

试题编号	CONNECT	试题名称	Find a special connected block
题目大意	给定一个 $n \times m$ 的棋盘，棋盘上每一个点有一个 -1 到 nm 之间的数。选择每个点有一个代价，你需要选择一些点使得这些点包含了 k 个不同的正整数，且不能选择有 -1 的点，且选择的点必须连通。问最小代价。		
算法讨论	首先考虑最大的数只有 k 的情况。此时必须选择每种正整数。可以记录 DP 状态 $f[s][i]$，表示当前 k 个整数的选择状态为 s，且包含位置 i 的最小代价。 转移时如果位置 i 包含一个当前在 s 中选择的数，则可以从去掉这个数的状态中转移过来。此外还可以通过两个包含 j 的连通块合并得到，可以枚举 s 的子集转移。还可以通过已经包含 s 的状态中加入一些点得到，此时可以用类似最短路算法的方式转移，每次用一些状态更新另一些状态的答案。这样时间复杂度为 $O(nm3^k + n^2m^22^k)$。 现在考虑最大数有 nm 的情况。如果将这些数随机映射到 k 的范围内，则原来 k 个不同的数仍然不同的概率为 $\frac{k!}{k^k}$。当 $k = 7$ 时大约为 6.12×10^{-3}。这样可以多次随机，每次对映射之后的问题求出最优解，再取所有情况中最优的。如果取随机次数 $S = 500$，则 $k = 7$ 时正确概率有 0.95，可以通过本题。		
时空复杂度	时间复杂度 $O(S(nm3^k + n^2m^22^k))$，空间复杂度 $O(nm2^k)$。		

试题编号	LEBOXES	试题名称	Little Elephant and Boxes
题目大意	有 n 个盒子，每个盒子里有百分之 p_i 的概率有 v_i 的钱，否则有一个钻石。有 m 个物品，第 j 个物品需要 c_j 的钱和 d_j 个钻石。问最多能够购买物品数量的期望值。		
算法讨论	首先预处理出用 i 个钻石购买 j 个物品所需要的最少的钱。然后将盒子分为两部分，每部分内枚举所有情况，得到对应概率，并按照得到的钻石数量分类。 枚举能够购买的物品数量和两部分得到的钻石数量，则可以知道需要的钱。将两部分的情况按照得到的钱排序，可以线性扫描两个数组得到总概率。		
时空复杂度	时间复杂度 $O(m^2n + mn2^n)$，空间复杂度 $O(mn + 2^n)$。		

试题编号	TICKETS	试题名称	Selling Tickets
题目大意	有 m 个人，每个人对应一个数对 (a_i, b_i)。对于一些人，你需要给每个人分配一个 1 到 n 之间的整数使得这个整数是 a_i 或 b_i，且所有人对应的整数互不相同。求最大的 k 使得任意选出 k 个人一定可以找到一个分配方案。		
算法讨论	显然只需要选出最少的人使得没有分配方案。将 1 到 n 的整数看作点，人可以选择的两个整数看作边。注意到如果一些边连接的总点数小于边数，则一定无解。而当边数不超过点数且连通时最多只会有一个环，容易构造出一种分配方案。因此最小的分配方案一定满足边数比点数大 1。 注意到边数比点数大 1 的连通图一定有两个环，可以分几种情况考虑。如果这两个环有公共边，则考虑公共边区间的两个端点，相当于选出这两个点之间的三条不相交路径。可以枚举第一条边，再要求接下来的路径不经过这条边，记录到每个点的最短的三条路径，可以得到这一部分的答案。 如果两个环没有公共边，可以看作在两个环上各选择一个点，用最短路连接这两个环。可以预处理出经过每个点的最小环，再枚举两个点得到答案。注意有可能在同一个点经过两个环，因此还要预处理次小环。		
时空复杂度	时间复杂度 $O(m^2)$，空间复杂度 $O(n^2)$。		

试题编号	COOLNUM	试题名称	Cool Numbers
题目大意	对于一个正整数,令 s 为它的数位和。称一个数 n 为 Cool Number，当且仅当可以选出不超过三位，数位和为 d，满足 $n \mid (s-d)^d$。给定正整数 N，问不超过 N 的最大的 Cool Number 和超过 N 的最小的 Cool Number。		
算法讨论	显然只有最多三位非零的数一定是 Cool Number。剩下的 Cool Number 很少，可以搜索得到。注意到此时 $s-d>0$，则 $n \le (s-d)^d$。由于 $d \le 27$，可以发现 n 最多有 76 位，即 $s \le 684$。注意到对于 n 的质因数分解中的每一项 p^c，在 $s-d$ 中一定有一项 $p^{\lceil \frac{c}{d} \rceil}$，可以枚举所有可能的 n 判断。判断是否合法时枚举 d 判断是否有合法的 $s-d$，再枚举是否可以选出几位数之和为 d 。 对于每组询问，最多三位非零的答案容易贪心得到，再在求出的特殊 Cool Number 中二分得到答案。		
时空复杂度	时间复杂度 $O(\sum \log N)$，空间复杂度 $O(\log N)$。		

试题编号	MATCH	试题名称	Expected Maximum Matching
题目大意	给定一个二分图，每条边有一定概率存在。问期望最大匹配。		
算法讨论	记录左边每个点集是否能和右边得到完备匹配。由霍尔定理，这样的点集要求每个子集都与右边的连接的不同点的个数至少为子集大小。设 DP 状态 $f[i][j]$ 表示右边加入了 i 个点，左边点集的状态为 j 的概率。每次转移时枚举新加入的点与左边每个点的连通状态，再考虑点集状态的转移。 注意到每次新加入一个右边的点时，对于每个左边连接的点，相当于所有之前不包含这个点的合法点集加上这个点后会仍然合法。这样可以得到新的点集状态。 可以发现可能转移出的点集状态很少，可以通过本题。		
时空复杂度	时间复杂度 $O(m2^n f(n))$，空间复杂度 $O(2^n f(n))$，其中 $f(n)$ 为可能得到的点集状态个数。		

试题编号	DGCD	试题名称	Dynamic GCD
题目大意	给定一棵树，每个点上有一个数。有一些询问，每次会询问一条路径上的所有数的 gcd，或将一条路径上的所有数增加一个值。		
算法讨论	首先进行树链剖分，则每个询问变为在一些区间上的操作。将序列差分，则一个区间的 gcd 为第一个数与区间内所有差的 gcd 的 gcd，修改操作变为修改两个数。可以用线段树解决本题。		
时空复杂度	时间复杂度 $O(q\log^2 n)$，空间复杂度 $O(n)$。		

试题编号	EST	试题名称	Equivalent Suffix Tries
题目大意	给定一个只包含小写字母的字符串，问有多少只包含小写字母的字符串和这个串有同构的后缀 Trie。		
算法讨论	注意到两个后缀 Trie 同构需要叶子个数相同。在后缀 Trie 中如果一个后缀没有对应叶子，则比它长度小的后缀都没有对应叶子。因此需要找到最长的后缀使得它是另一个后缀的前缀。注意到每种不同的字母都在对应叶子结点的后缀部分出现过。这些位置的字符的相等关系一定在任何合法串中都相同，因此可以看作这些位置一定不变，再将答案乘以排列数。 注意到剩下部分一定是一个后缀的前缀，可以发现最多只有 $O(n)$ 种方案。如果要求两个后缀 Trie 同构，则任意两个后缀的 LCP 要保持不变。这等价于已经确定部分的后缀和这一部分最后一个后缀的 LCP 保持不变。找到 LCP 最大的后缀，则可以继续确定一些字符，同时可以发现这些 LCP 中比最大值小的一定满足不变，相等的可能会更大。这样相当于在下一位有一些字符不能选择，否则这些 LCP 会比原来的值大。 这样可以在 $O(1)$ 时间内检验一种方案是否合法。枚举这个后缀是哪个后缀的前缀，可以在 $O(n)$ 时间内找到所有合法方案。但是这样可能会得到重复解，需要去重。 去重可以用字符串哈希。现在需要快速求出选择作为一个后缀的前缀时这个串的哈希值。注意到在可以用这个后缀确定的一部分之后一定会循环出现这个串。可以预处理出一个周期内的哈希值，再得到整个串的哈希值。		
时空复杂度	时间复杂度 $O(n\log n)$，空间复杂度 $O(n)$。		

试题编号	MAGIC	试题名称	Two Magicians
题目大意	给定一个无向图。一开始在 1 和 2 各有一个人。两个人轮流行动，每次行动需要加入一条边，并可以选择传送到另一个点。每个人传送次数最多 p 次。行动后和对方在同一个连通块中就输了。问谁有必胜策略。		
算法讨论	注意到加入一条边要么合并两个连通块，要么连接同一个连通块中的点。显然如果不合并连通块则所有边等价，只需要记录这种边的个数。由于两个人可以各自加上一条这样的边，只需要记录这种边的个数的奇偶性。这样连通块的大小也可以变为连通块的大小的奇偶性。只需要记录大小为奇数和偶数的连通块的个数，当前两个人所在连通块的大小的奇偶性，两个人传送的次数。这样状态数为 $O(n^2p^2)$。 注意到连接两个连通块的边按照奇偶性可以分为三种情况。如果连接两个偶数大小的连通块，相当于减少一个偶数大小的连通块并增加奇数条块内的边。连接一个偶数大小的连通块和一个奇数大小的连通块会有同样的效果。因此如果连接的连通块中有一个大小为偶数，且连通块至少有三个，则可以选择一种不连接两个人所在连通块的方案。因此在这种情况下不需要传送。 如果连接两个奇数大小的连通块，且没有其它奇数大小的连通块，则这种情况下需要传送。但是这一步之后不会有奇数大小的连通块，因此整个游戏中最多只会需要一次传送。可以将状态数变为 $O(n^2)$。 可以发现当偶数大小的连通块数量较大时，答案与减少一个偶数大小的连通块后的答案相同。当奇数大小的连通块数量较大时，答案有周期性。这样只需要预处理小范围内的答案，可以解决本题。		
时空复杂度	时间复杂度 $O(m)$，空间复杂度 $O(1)$。		

试题编号	GTHRONES	试题名称	A Game of Thrones
题目大意	有 n 种数，每个数 u_i 有 c_i 个。两个人进行博弈。先手选择一个数，之后两个人轮流选择与之前的数的质因数分解中只相差一个质数的数，不能选择已经选择过的数。不能选择的人输。问谁有必胜策略。		
算法讨论	首先可以对每一对数判断能否从一个数移动到另一个数。需要判断一个数是否为质数，可以用 Miller-Rabin 算法解决。之后得到一个图，两个人可以轮流移动到相邻的点。当这个图有完美匹配时后手胜，否则先手胜。 注意到这个图是二分图，可以用二分图最大匹配。由于每个数可以出现多次，可以用最大流解决，将边的容量设定为这个数出现的次数。		
时空复杂度	时间复杂度 $O(n^4)$，空间复杂度 $O(n^2)$。		

试题编号	KNGHTMOV	试题名称	Knight Moving
题目大意	给定一个无限大的棋盘，上面有一些障碍。你从 $(0,0)$ 开始，每次可以从 (u,v) 移动到 $(u+A_x, v+A_y)$ 或 $(u+B_x, v+B_y)$。问你有多少种方案走到 (X,Y)，或无限多种。		
算法讨论	如果移动的两种方法线性相关，则只能在一条直线上移动。此时可以变为一维问题解决。注意到如果一条路径上的一个点可以走出一个环，则有无限多种方案。如果可以走出环，则一定可以在三倍坐标范围内走出环，因此只要考虑范围内的情况。可以 DFS 得到答案。 如果移动的两种方法线性无关，则可以确定两种移动方法的使用次数。此时可以变为只能向上或向右一格的情况解决。可以容斥计算出不经过障碍的方案数，容斥时可以递推解决。		
时空复杂度	时间复杂度 $O(s^2+k^2)$，空间复杂度 $O(s^2+k)$，其中 s 为坐标范围。		

试题编号	PARADE	试题名称	Annual Parade
题目大意	给定一个带权无向图，你需要选择一些路径。对于每条路径经过的每条边，你需要付出边权的代价。如果一个点没有被覆盖，需要付出 C 的代价。如果一条路径不是环，需要付出 C 的代价。有若干询问，每次给定 C，问最小代价。		
算法讨论	注意到一条路径可以分为若干段，每一段的端点为之前没有被覆盖的点。每得到一段可以减小 C 的代价。注意到这样同时处理了不是环的代价。这样可以用最小费用流解决。将每个点拆为两个点，一段路径可以从一个点中的第一个流到另一个点的第二个。每个点最多只能被使用一次。如果每次只流出一个单位流量，可以得到每个流量需要的额外代价。对于每个询问可以二分得到答案。		
时空复杂度	时间复杂度 $O(n^2m+k\log n)$，空间复杂度 $O(m)$。		

<table>
<tr><td>试题编号</td><td>MAXCIR</td><td>试题名称</td><td>Max Circumference</td></tr>
<tr><td>题目大意</td><td colspan="3">给定三个点 A，B，C，有 n 个操作 (x_i, y_i)。每个操作会将 A 变为 $(x_A + x_i, y_A + y_i)$。你可以选择不超过 k 个操作，使得三角形 ABC 的周长最大。</td></tr>
<tr><td>算法讨论</td><td colspan="3">显然只需要让 $AB + AC$ 最大。注意最优解一定可以通过选择在某个方向上达到最大得到。这样可以枚举这个方向，求出这个方向上的最优解，再取其中最优的。
注意到在一个方向上的最优解一定时按照移动向量在这个方向上的投影排序后取最大的 k 个或所有大于 0 的操作。这样所有使得投影的排序相同的方向会得到同样的解。如果按照极角序枚举方向，则只有当枚举的方向经过两个点连线垂直的方向时会导致投影的排序发生变化。同样，只有当枚举的方向经过一个操作向量的垂直方向时会导致这个操作是否大于 0 的状态发生变化。可以将所有这样的方向排序，则相邻两个方向之间的一段等价。在经过两段的分界处时相当于在排序中交换两项，或一个操作是否大于 0 的状态发生变化。可以直接维护，同时维护最大的 k 个或所有大于 0 的操作的和。
由于本题对精度要求很高，在实现时需要分开存储整数部分和小数部分。在计算 $\sqrt{x}$ 的小数部分时可以用 $\sqrt{x} - d = \frac{x-d^2}{\sqrt{x}+d}$ 实现。</td></tr>
<tr><td>时空复杂度</td><td colspan="3">时间复杂度 $O(n^2 \log n)$，空间复杂度 $O(n^2)$。</td></tr>
</table>

试题编号	CIELHACK	试题名称	Ciel and password cracking
题目大意	你需要破解一个密码。这个密码有 k 部分，你需要找到 k 个地方，每个地方破解一部分。每一部分是一个不超过 n_i 的正整数。 在第 j 个地方破解密码时需要选择使用不超过 p_j 台电脑，之后每台电脑独立破解密码，在 t_j 的时间内枚举一个密码，且同一台电脑不会枚举重复的密码，当有一台电脑破解出密码时停止。 此外选择的电脑需要传输信息。一开始只有 1 号电脑有信息，之后已经有信息的电脑会按顺序给其它电脑信息，传输信息需要的时间服从期望为 s_j 的指数分布。当一台电脑得到信息时所有尝试给这台电脑信息的电脑会停止并继续给下一台电脑传递信息。 第 j 个地方在 x_j 的位置。你一开始在 $x=0$，且在 x 上移动速度为 v。问你破解出所有密码并回到 $x=0$ 需要的最短期望时间。		
算法讨论	首先考虑用 m 台电脑破解大小不超过 n 的期望时间。在传递信息的部分所有方案都等价，因为当有 i 台电脑在传递信息时，接下来最快传递成功的期望时间为 $\frac{s}{i}$。这样这一部分的用时为 $\sum_{i=1}^{m-1}\frac{s}{i}=sH_{m-1}$。在破解密码的部分，每台电脑都尝试至少 k 个密码的概率为 $(\frac{n-k+1}{n})^m$，因此期望用时 $\frac{1}{n^m}\sum_{i=1}^{n}i^m$。 现在考虑如何快速计算这些值。由于 $H_n=\log n+C+\frac{1}{2n}+O(n^{-2})$，可以预处理小范围内的 H_n 并计算出 C，就能快速计算 H_n。要计算 $\sum_{i=1}^{n}i^m$，可以用公式得到 $\frac{1}{m+1}\sum_{i=0}^{m}\binom{m+1}{i}B_in^{m+1-i}$，其中 B_i 为伯努利数。当 $\frac{m+1}{n}$ 较小时可以只取前几项近似。而当 $\frac{m+1}{n}$ 较大时 $(\frac{i}{n})^m$ 可以只取后几项近似。这样可以快速求出用 m 台电脑破解大小不超过 n 的密码的期望时间。 注意到这个时间是可以三分的，这样可以三分使用几台电脑，就能得到在某个地方破解某个密码的最短期望时间。之后可以发现在 x 上移动的时间只与最大和最小的 x 有关，同时当有一边没有选择时要特殊处理。容易在 $O(kc)$ 时间内 DP 得到答案。		
时空复杂度	时间复杂度 $O(kc\log n)$，空间复杂度 $O(kc)$。		

试题编号	COUNTARI	试题名称	Arithmetic Progressions
题目大意	给定一个长度为 n 的序列，问有多少长度为 3 的子序列为等差数列。		
算法讨论	将序列每 k 个元素分一块。枚举中间元素所在的块，分几种情况考虑。 如果三个元素都在这一块，可以枚举剩下元素中的一个，计算出剩下元素的值，维护值为每个数的数的个数，就可以得到答案。当这一块中有两个元素时可以类似枚举另一个元素，得到剩下元素的值，再求出答案。 当这一块中只有一个元素时，左边和右边元素之和为这一个元素的两倍。可以将左边和右边每个数出现的次数做卷积，再枚举中间元素得到答案。 这样时间复杂度为 $O(\frac{n^2}{k}+ks\log s)$，其中 s 为数的范围。取 $k=\frac{n}{\sqrt{s\log s}}$ 时达到最优。		
时空复杂度	时间复杂度 $O(n\sqrt{s\log s})$，空间复杂度 $O(s)$。		

试题编号	MARTARTS	试题名称	Martial Arts
题目大意	给定一个带权二分图，每条边有两个权值。你需要求出第一权值最大，相同时第二权值最大的匹配。匹配的权值计算为匹配中所有边的权值之和减去第一权值最大，相同时第二权值最小的边权。		
算法讨论	枚举匹配中被删去的边权，则按照从小到大的顺序加入，需要维护只使用前面一部分边，且有两个点被强制连接的最大权匹配。对于没有加入的边可以令权值为负无穷大，对于强制选择的边可以令权值为正无穷大。这样需要在修改边权的情况下维护最大权匹配。 在 KM 算法中，修改两点之间的权值只需要将其中一个点断开匹配，再重新寻找增广路。这样每次修改的时间复杂度为 $O(n^2)$，可以通过本题。		
时空复杂度	时间复杂度 $O(n^4)$，空间复杂度 $O(n^2)$。		

试题编号	DIFTRIP	试题名称	Different Trips
题目大意	给定一棵树，每个点上有一个标号为它的度数。定义两条路径相似，当且仅当路径上经过的点的标号相同。问有多少条从一个点往上走到另一个点的不相似的路径。		
算法讨论	求出这棵树的前缀数组，则答案为所有可能的路径数减去相邻两个位置之间相似的个数和。求树的前缀数组与倍增求后缀数组相似。		
时空复杂度	时间复杂度 $O(n\log n)$，空间复杂度 $O(n)$。		

<table>
<tr><td>试题编号</td><td>QPOLYSUM</td><td>试题名称</td><td>Quasi-Polynomial Sum</td></tr>
<tr><td>题目大意</td><td colspan="3">给定 D 次多项式 $P(x)$ 的值 $P(0), P(1), \dots, P(D)$。求出 $(\sum_{i=0}^{n-1} P(i)Q^i) \bmod m$ 的值。m 与 2 到 $D+14$ 之间的所有整数互质。</td></tr>
<tr><td>算法讨论</td><td colspan="3">
由于 P 是一个 D 次多项式，得到

$$\sum_{i=0}^{D+1} (-1)^i \binom{D+1}{i} P(n-i) = 0$$

令 $F(n) = P(n)Q^n$，则

$$\sum_{i=0}^{D+1} (-1)^i \binom{D+1}{i} F(n-i)Q^i = 0$$

即

$$F(n) = \sum_{i=0}^{D} (-1)^i \binom{D+1}{i+1} F(n-i-1)Q^{i+1}$$

令 $S(n) = \sum_{i=0}^{N} F(i)$，则

$$S(n) = \sum_{i=0}^{D} (-1)^i \binom{D+1}{i+1} S(n-i-1) + \sum_{i=0}^{D} (-1)^i \binom{D+1}{i} S(D-i)Q^i$$

令 $c = \sum_{i=0}^{D} (-1)^i \binom{D+1}{i} S(D-i)Q^i$，则 c 容易求出。递归展开得到

$$S(n) = \sum_{i=0}^{D} (-1)^i \binom{n-D-1+i}{i} \binom{n}{D-i} S(D-i)Q^{n+i} + c \sum_{i=0}^{n-D-1} \binom{D+i}{D} Q^i$$

前一部分容易在 $O(D + \log n)$ 时间内求出，考虑 $G(n, D) = \sum_{i=0}^{n-D-1} \binom{D+i}{D} Q^i$。容易得到 $G(n, 0)$ 的值。可以得到

$$G(n, D) = \frac{1}{Q-1}(\binom{n}{D} Q^{n-D} - G(n, D-1))$$

当 m 与 $Q-1$ 互质时可以递推求出。这样可以将 m 分为与 $Q-1$ 互质的一部分和剩下的一部分。注意到剩下一部分 m 存在 k 满足 $m \mid (Q-1)^k$。由题目可得 $k \le 14$。这样只需要求出 $G(n, D) \bmod (Q-1)^k$。可以发现

$$G(n, D) \equiv \binom{n}{D+1} Q^{n-D-1} \pmod{Q-1}$$

同样可以类似求出 $G(n, D) \bmod (Q-1)^k$。
</td></tr>
<tr><td>时空复杂度</td><td colspan="3">时间复杂度 $O(D + \log n + k)$，空间复杂度 $O(D)$。</td></tr>
</table>

试题编号	ANDOOR	试题名称	A New Door
题目大意	给定一些圆和一个矩形，求出这些圆的并集在矩形内部的周长。		
算法讨论	可以类似圆并算法，枚举每个圆，再得到这个圆的周长中没有被覆盖的部分。对于其它的圆和矩形，求出与这个圆的交点，则会覆盖一个区间。将区间端点排序后可以得到答案。注意边界情况的处理。		
时空复杂度	时间复杂度 $O(n^2 \log n)$，空间复杂度 $O(n)$。		

试题编号	CUCUMBER	试题名称	Cucumber Boy and Cucumber Girl
题目大意	给定 b 个 $n \times n$ 的矩阵，问有多少对矩阵 A 和 B，满足 A^TB 中选择 n 个元素使得每一行每一列有一个元素且至少选择一个奇数的方案数为奇数。		
算法讨论	显然只需要在模 2 意义下计算。令 S 为 $n \times n$ 的每个数为 1 的矩阵，则在 A^TB 中选择 n 个元素使得每一行每一列有一个元素且至少选择一个奇数的方案数为 $n! - \mathrm{perm}(S + A^TB)$。$n = 1$ 时特判，当 $n > 1$ 时方案数为奇数等价于 $\mathrm{perm}(S + A^TB) \equiv 1 \pmod 2$。 由于 $\mathrm{perm}(A) \equiv \det(A) \pmod 2$,只需要判断 $\det(S+A^TB)$ 的奇偶性。可以将给定的每个矩阵增加一行,值全部为 1。这样 $S+A^TB = A'^TB'$ 由 Cauchy-Binet 公式得 $\det(A'^TB') = \sum_{i=1}^{n+1} \det(A'_i)\det(B'_i)$，其中 A'_i 为 A' 去掉第 i 行后的矩阵。这样只需要求出所有 $\det(A'_i)$ 的奇偶性。这相当于求出 A' 中去掉哪些行可以使得剩下的矩阵满秩。 对每个 A' 进行高斯消元。如果矩阵的秩小于 n，则所有 A'_i 均为 0。否则去掉没有被消去的一行显然满秩。同时去掉这一行中剩下的列对应的行也可以满秩。这样可以求出 A'_i。 求出 A'_i 后可以枚举两个矩阵用位运算判断是否合法。		
时空复杂度	时间复杂度 $O(bn^2 + b^2)$，空间复杂度 $O(b)$。		

试题编号	ROC	试题名称	Room Corner
题目大意	给定一个图形表示一个房间，每个角落有一个人。在每个时刻可以有一些人交换位置，两个人分别沿着房间边界移动到另一个人的位置。每个人同一时刻只能和一个人交换位置。有一些询问，每次询问两个人相遇需要的最短时间。		
算法讨论	首先求出房间内的所有角落和沿着房间边界的顺序，和相邻两个角落之间的距离。这样问题变为一个环。对于每组询问，枚举两个人之间在哪个方向相遇，则变为区间上的问题。 注意到只有在倒数第二次交换位置时不能做到两个人同时移动,可以枚举最后一次交换位置是在哪两个角落之间。容易在 $O(1)$ 得到这种情况下的答案。		
时空复杂度	时间复杂度 $O(tn)$，空间复杂度 $O(n)$。		

试题编号	QUERY	试题名称	Observing the Tree
题目大意	给定一棵树，每个点有一个权值，一开始为 0。有一些询问，每个询问会修改一条路径上的权值，对于路径上经过的第 k 个点增加 $A+(k-1)B$ 的权值，或询问一条路径上的权值和，或将权值恢复到第 x 个询问后的状态。强制在线。		
算法讨论	首先进行树链剖分，然后每条路径变为一些区间。可以求出每个区间中路径的方向，和区间端点处是路径中第几个点。对于一个区间的修改，可以看作对位置为 i 的点增加 $A'+B'i$ 的权值。对于每个区间，可以在线段树上修改这个区间。由于有恢复之前状态的操作，需要用可持久化线段树维护。		
时空复杂度	时间复杂度 $O(n\log^2 n)$，空间复杂度 $O(n\log^2 n)$。		

试题编号	LECOINS	试题名称	Little Elephant and Colored Coins
题目大意	有 n 种硬币，每种硬币有无限个。每种硬币有一个面值和一个颜色，问在拼出总面值为 S 的方案中最多可以使用多少不同的颜色。		
算法讨论	对于一种方案，可以看作先选择一些颜色不同的硬币，再任意选择一些硬币，选择的颜色数为第一部分选择的硬币数量。 首先求出第二部分中哪些面值可以拼出。由于每种硬币有无限个，可以按照总面值模某个硬币面值分类。这样每一类中有一个面值可以拼出则这一类中所有更大的面值也可以拼出。只需要求出每一类中最小的可以拼出的面值。对于每种其它的硬币，可以将一类中的答案转移到另一类中。这样可以用最短路求出每一类中的答案。 对于第一部分的答案，可以在第二部分答案上 DP，求出选择一定数量的颜色时每一类的答案。		
时空复杂度	时间复杂度 $O(nv\log v+n^2v+qn)$，空间复杂度 $O(v)$。		

<table>
<tr><td>试题编号</td><td>CHANGE</td><td>试题名称</td><td>Making Change</td></tr>
<tr><td>题目大意</td><td colspan="3">有 n 种硬币，每种硬币有无限个。给出每种硬币的面值，问用这些硬币拼出总面值为 c 的方案数。硬币的面值两两互质。</td></tr>
<tr><td>算法讨论</td><td colspan="3">考虑拼出面值的方案数的生成函数 $f(x)$。
$$f(x)=\prod_{i=1}^{n}\frac{1}{1-x^{d_i}}=\frac{1}{(1-x)^n}\prod_{i=1}^{n}\prod_{j=1}^{d_i-1}\frac{1}{1-w_{d_i}^{j}x}$$
其中 w_n 是单位复根。进行部分分式分解得到
$$f(x)=\frac{A(x)}{(1-x)^n}+\sum_{i=1}^{n}\sum_{j=1}^{d_i-1}\frac{B_{i,j}}{1-w_{d_i}^{j}x}$$
其中 $A(x)$ 是一个不超过 $n-1$ 次的多项式，$B_{i,j}$ 为常数。考虑求 $B_{p,q}$。得到
$$(1-w_{d_p}^{q}x)f(x)=\frac{(1-w_{d_p}^{q}x)A(x)}{(1-x)^n}+\sum_{i=1}^{n}\sum_{j=1}^{d_i-1}\frac{B_{i,j}(1-w_{d_p}^{q}x)}{1-w_{d_i}^{j}x}$$
令 x 无限趋近 $w_{d_p}^{-q}$，则得到
$$B_{p,q}=\frac{1}{d_p}\prod_{i\neq p}\frac{1}{1-w_{d_p}^{-qd_i}}$$
考虑到在部分分式中的一项 $\frac{C_i(x)}{\sum_{j=0}^{d_i-1}x^j}$，在 $C_i(x)$ 中 x^m 的系数为 $\sum_{j=1}^{d_i-1}B_{i,j}\frac{1-w_{d_i}^{(m+1)j}}{1-w_{d_i}^{j}}$。即
$$\frac{1}{d_i}\sum_{j=1}^{d_i-1}\frac{1-w_{d_i}^{(m+1)j}}{1-w_{d_i}^{j}}\prod_{p\neq i}\frac{1}{1-w_{d_i}^{-jd_p}}$$
注意到求和的部分是一个关于 $w_{d_i}^{j}$ 的函数且这个数一定满足 $\sum_{j=0}^{d_i-1}x^j=0$。这个函数为 $\frac{1-x^{m+1}}{1-x}\prod_{p\neq i}\frac{1}{1-x^{-d_p}}$，可以将 $\frac{1-x^{m+1}}{1-x}$ 转化为 x^m，再将所有 $-d_p$ 变为模 d_i 意义下的值。这样变为 $x^m\prod_{p\neq i}\frac{1}{1-x^{u_p}}=\frac{x^m}{(1-x)^{n-1}}\prod_{p\neq i}\frac{1}{\sum_{j=0}^{u_p-1}x^j}$。
首先对 $\frac{x^m}{(1-x)^{n-1}}$ 的部分求和。当 $n=1$ 时可以直接求出。假设已经求出了 $n=k$ 时的答案，则注意到 $\frac{x^m}{(1-x)^n}-\frac{x^{m+1}}{(1-x)^n}=\frac{x^m}{(1-x)^{n-1}}$，对 $n=k+1$ 的答案求差分可以得到 $n=k$ 的答案。又由于 $\sum_{m=0}^{d_i-1}\frac{x^m}{(1-x)^n}=0$，可以先求出 $n=k$ 时答案的前缀和，再调整得到 $n=k+1$ 时的答案。
对于剩下的部分，每次加入一个新的 p。由于 $\sum_{j=0}^{d_i-1}x^j=0$，只需要求出所除的式子对这个多项式的逆元。这个逆元容易构造求出。这样可以 $O(d)$ 时间内求出一个 m 的答案。对于剩下的答案可以每次加入一个并删除一个递推得到。
对于 $A(x)$ 的部分，可以用 DP 求出前 n 项的答案，再插值得到。</td></tr>
<tr><td>时空复杂度</td><td colspan="3">时间复杂度 $O(n^2d)$，空间复杂度 $O(nd)$。</td></tr>
</table>

<table>
<tr><td>试题编号</td><td>STRQUERY</td><td>试题名称</td><td>String Query</td></tr>
<tr><td>题目大意</td><td colspan="3">给定一个字符串，要求支持在字符串的开头、中间、结尾插入或删除一个字符。其中字符串的中间定义为第 $\lfloor \frac{l}{2} \rfloor$ 个字符，其中 l 为字符串长度。有一些询问，每次询问一个串在这个字符串中出现的次数。</td></tr>
<tr><td>算法讨论</td><td colspan="3">首先考虑只有在开头修改的情况。可以用后缀平衡树实现所有操作。
考虑可以在两段修改的情况。可以将串分为左右两部分，分别维护两个方向的后缀平衡树。当一个部分为空时如果需要继续在这个方向删除，则可以重新均匀分为两部分重建。询问时在两边分别询问，再求出边界处的串用 KMP 算法匹配。
考虑原问题，可以将串均匀分为两部分，每一部分维护一个可以在两段修改的结构。在每次修改后如果不满足长度均匀，则可以将一个部分的最后一个字符删除并插入到另一个部分的开头。询问时和上面情况相似。</td></tr>
<tr><td>时空复杂度</td><td colspan="3">时间复杂度 $O(q \log q)$，空间复杂度 $O(q)$。</td></tr>
</table>

<table>
<tr><td>试题编号</td><td>INVBINCF</td><td>试题名称</td><td>Inverse Binomial Coefficient</td></tr>
<tr><td>题目大意</td><td colspan="3">给定 n，求出最小的满足 $0 \le k < 2^n$ 的整数 k 使得 $\binom{2^n-1}{k} \bmod 2^n = r$，或输出无解。</td></tr>
<tr><td>算法讨论</td><td colspan="3">考虑去掉所有 2 质因子之后的情况，可以发现
$$\binom{2^n-1}{k} = \frac{f(2^n-1)}{f(k)f(2^n-1-k)}\binom{2^{n-1}-1}{\lfloor \frac{k}{2} \rfloor}$$
其中 $f(n)$ 表示不超过 n 的所有奇数的乘积。显然当 r 为偶数时无解。可以证明对于所有奇数 r 一定有解。如果已经求出 $n = k$ 时的解 x，按照 x 的奇偶性讨论，可以得到 $n = k + 1$ 时的两种可能答案。只需要判断一个答案是否合法。这需要快速求出 $f(n)$ 的值。
如果需要求 $f(x)$，则考虑 x 的二进制表示，可以将不超过 x 的奇数分为一些段。对于最高位对应的一段可以预处理出来。对于剩下的段可以看作需要求出 $p + s$ 的乘积，其中 p 可以取不超过 2^i 的所有奇数。这样可以按照展开后 s 的指数分类，每一类中需要求出不超过 2^i 的所有奇数中选择 j 个数的乘积的和。这样对于每次询问 $f(x)$ 可以在 $O(n^2)$ 时间内得到答案，需要 $O(n^3)$ 时间。注意到在考虑 i 这一位时，s 一定是 2^{i+1} 的倍数，因此只需要求 $\frac{n}{i}$ 项，可以优化到 $O(n^2 \log n)$。
还需要预处理在不超过 2^i 的所有奇数中选择 j 个数的乘积的和。注意到如果已经求出 $i-1$ 的答案，可以用枚举在两部分数中各选择了多少个求出答案。但是在求这个答案时需要用到之前的 $j + \frac{n}{i}$ 部分的答案，因此需要对每个 i 预处理出 $O(n \log n)$ 个 j 的答案，时间复杂度为 $O(n^3 \log^2 n)$。注意到随着 i 增大，需要维护的 j 数量会减少，可以用 $O(\sum_{i=1}^{n} (n + \sum_{j=i}^{n} \frac{n}{j})^2)$ 的时间预处理出答案。可以证明是 $O(n^3)$ 的。</td></tr>
<tr><td>时空复杂度</td><td colspan="3">时间复杂度 $O(n^3 + tn^2 \log n)$，空间复杂度 $O(n^2)$。</td></tr>
</table>

试题编号	QTREE	试题名称	Queries on tree again!
题目大意	给定一个有 n 个点 n 条边的简单无向连通图，保证图中环长度为奇数。每条边有一个权值。有一些询问，每个询问可以将两个点之间的最短路上的边权取负，或询问两个点之间的最短路上边权的最大权值的连续一段，可以为空。		
算法讨论	首先找到图中的环，去掉环上的一条边，得到一棵树。对这棵树进行树链剖分。对于两个点之间的最短路，如果两个点在环上同一个点的子树内，则为这棵树上两个点之间的路径，否则考虑沿着环的哪个方向距离较小，最短路为树上两个点之间的路径或从一个点走到环的第一个点，再走到环的最后一个点，再走到另一个点的路径。可以用树链剖分将路径分为 $O(\log n)$ 部分。 对于路径的每个部分，可以在线段树上维护答案。对于询问操作，需要维护一段内的答案和最大权值的前缀和后缀。对于修改操作，由于需要翻转符号，还需要维护对应的最小值。		
时空复杂度	时间复杂度 $O(n\log^2 n)$，空间复杂度 $O(n)$。		

试题编号	CPP	试题名称	Chef Protection Plan
题目大意	给定一个格式串和一些询问串，问每个询问串是否符合这个格式串，或判断格式串不合法。 每个格式串可以有这些部分： `x to y` 表示一个从 `x` 到 `y` 之间的字符，其中 `x` 和 `y` 同时为数字或大写字母，且 `x` 在 `y` 字典序之前。 `x or y` 表示匹配两个格式串中任意一个，其中 `x` 和 `y` 为格式串。 `x N times` 表示重复格式串 `x`，`N` 次，其中 `x` 为格式串，`N` 为 2 到 12 之间的整数。 `x optional` 表示匹配 `x` 或空串，其中 `x` 为格式串。 `digit` 匹配一个数字。`letter` 匹配一个字母。 `exactly N digits` 和 `exactly N letters` 分别表示匹配 `N` 个数字或字母。`upto N digits` 和 `upto N letters` 分别表示匹配不超过 `N` 个数字或字母。其中 `N` 为 2 到 12 之间的整数。 将两个格式串连接可以得到另一个格式串，表示匹配能够分为两部分，分别被两个格式串匹配的串。 可以用括号改变优先级。 要求合法的格式串不能匹配长度超过 2000 的串。		
算法讨论	首先对格式串进行语法分析，可以得到格式串的信息。注意到格式串中所有的部分都可以用正则表达式表示，可以将格式串转化为等价的正则表达式，再用正则表达式解决。		
时空复杂度	时间复杂度 $O(t(f(\|F\|)+qg(\|F\|,\|S\|)))$，空间复杂度 $O(h(\|F\|))$，其中 $f(\|F\|),g(\|F\|,\|S\|),h(\|F\|)$ 为正则表达式匹配算法的复杂度。		

试题编号	TKCONVEX	试题名称	Two k-Convex Polygons
题目大意	给定 n 个正整数，选出 $2k$ 个数分为两组，每组 k 个数，使得每一组中的最大的数小于这一组数的和的一半，或输出无解。		
算法讨论	注意到如果有满足条件的 k 个数，则一定可以找到排序后连续的 k 满足条件。当 n 足够大时，如果没有满足条件的 k 个数，则最大的数将会超过范围，因此一定有解。这样当 n 足够大时可以任意选出合法的一组数，一定可以在剩下的数中选出另一组数。这样只需要考虑 n 较小的情况。 如果两组数在排序后的序列中不相交，则可以枚举第一组数，再判断是否有解。当两组数在排序后的序列中相交时，选择的 $2k$ 个数一定是连续的 $2k$ 个数。枚举这些数的位置，再枚举划分成两组的方案，判断是否合法。		
时空复杂度	时间复杂度 $O(nk + Sk\binom{2k}{k})$，空间复杂度 $O(n)$，其中 S 为保证有解的最小的 n。		

试题编号	SPMATRIX	试题名称	Count Special Matrices
题目大意	问有多少 $n \times n$ 的矩阵 A，满足对角线上元素为 0，对于所有 $i < j$ 满足 $A_{ij} = A_{ji}$ 且 $1 \le A_{ij} \le n-2$，对于所有 i, j, k 满足 $A_{ij} = \max(A_{ik}, A_{jk})$，且 1 到 $n-2$ 之间的每个数都出现至少一次。		
算法讨论	可以将 A 看作一个带权无向图的邻接矩阵。如果只考虑图中权值至少为 k 的边，则可以发现图中每一个连通块中的点两两有边。增加权值为 $k-1$ 的边可以看作将图中的一些连通块合并。 由于有 $n-2$ 种不同权值的边，一定有一个权值合并三个连通块或合并两次两个连通块，而剩下的权值合并两个连通块。当有 m 个连通块时，合并两个连通块的方案数为 $\frac{m(m-1)}{2}$，而合并三个连通块的方案数为 $\frac{m(m-1)(m-2)}{6}$，合并两次两个连通块的方案数为 $\frac{m(m-1)(m-2)(m-3)}{8}$。 如果权值为 k 时合并三个连通块或合并两次两个连通块，则可以得到总方案数为 $\frac{n!(n-1)!(3k+1)}{2^{n-1}(6k+6)}$。这样答案为 $$\frac{n!\,(n-1)!}{2^{n-1}}\sum_{k=1}^{n-1}\frac{3k+1}{6k+6} = \frac{n!\,(n-1)!}{2^{n-1}}(\frac{n}{2} - \frac{H_{n-1}+2}{3})$$ 其中 H_n 为调和数。这样可以得到一个 $O(n)$ 的算法，但是直接实现这个算法需要 $3n$ 次模意义下的乘法，会超时。 令 $f_n = n!\,H_n$，则 $f_0 = 0$，$f_n = nf_{n-1} + (n-1)!$。这样可以递推求出 f_n，再求出答案。这样只需要 $2n$ 次模意义下的乘法，可以通过本题。		
时空复杂度	时间复杂度 $O(n + t)$，空间复杂度 $O(n)$。		

试题编号	LYRC	**试题名称**	Music & Lyrics
题目大意	给定一些模式串和一些文本串，问每个模式串在所有文本串中出现的总次数。		
算法讨论	对模式串建出 AC 自动机，再对所有文本串匹配。		
时空复杂度	时间复杂度 $O(w\|P\| + n\|S\|)$，空间复杂度 $O(w\|P\|)$。		

试题编号	PRIMEDST	**试题名称**	Prime Distance On Tree
题目大意	给定一棵树，问任意选出两个不同的点，这两个点之间的距离为质数的概率。		
算法讨论	对树进行点分治，在每一层内需要考虑选出两个点深度之和为质数的方案数。可以对每个深度的点的个数进行卷积，得到距离为每个数的点对的个数。卷积可以用 FFT 实现。		
时空复杂度	时间复杂度 $O(n \log^2 n)$，空间复杂度 $O(n)$。		

试题编号	TWOROADS	**试题名称**	Two Roads
题目大意	给定 n 个点，求出两条直线，使得每个点到这两条直线的距离的较小值的平方和最小。输出这个最小值。		
算法讨论	如果只有一条直线,容易计算出最优解。注意到这个答案只与 $\sum x^2$，$\sum y^2$，$\sum xy$，$\sum x$，$\sum y$ 和 n 有关。 当有两条直线时，有一部分点到第一条直线的距离较小，另一部分点到第二条直线的距离较小。注意到这两部分的分割线是选择的两条直线夹角的角平分线，两条分割线一定垂直。这样可以考虑枚举这两条分割线，再对两部分分别求出答案。注意到如果求出的答案对应的分割和枚举的不同，可以忽略，因为这样的解一定不是最优解。 考虑如何枚举所有分割线。当固定一条分割线时，随着另一条分割线的移动，会产生 $O(n)$ 种分割方案。可以在移动的过程中维护与答案相关的值，这样可以在 $O(n)$ 时间内得到这些方案的答案。注意到只要所有点在第一条直线上的投影的顺序不变，得到的方案也不变。这样当第一条直线方向改变时，只有经过两个点连线垂直方向时会有变化。这样可以将所有可能的方向排序，再扫描所有方向。注意到每次只会交换两个点的位置，可以在 $O(1)$ 时间维护这个顺序。		
时空复杂度	时间复杂度 $O(n^3)$，空间复杂度 $O(n^2)$。		

试题编号	TMP01	试题名称	To Queue or not to Queue
题目大意	给定一个字符串，初始为空。有一些操作，每个操作可以在最后插入一个字符，或在开头删除一个字符。每个操作后询问当前字符串中本质不同的子串个数。		
算法讨论	考虑用后缀树解决。本质不同的子串为后缀树中所有边的长度和。对于插入操作，用后缀树的在线构造算法可以解决。对于删除操作，相当于在后缀树中删除一个后缀。可以删除这个后缀对应的结点，再不断删除度数为 0 的父亲结点。如果删除了后缀树构造算法中当前位置所在的边，则当前位置会对应一个叶子结点，可以用类似插入时的做法新建一个叶子结点。		
时空复杂度	时间复杂度 $O(q)$，空间复杂度 $O(q)$。		

试题编号	FN	试题名称	Fibonacci Number
题目大意	给定质数 p 和非负整数 c，满足 $p \mod 10$ 是完全平方数。求出最小的非负整数 n，使得 $f_n \equiv c \pmod p$，其中 f_n 为斐波那契数。		
算法讨论	由通项公式得到 $$f_n = \frac{\left(\frac{1+\sqrt{5}}{2}\right)^n + \left(\frac{1-\sqrt{5}}{2}\right)^n}{\sqrt{5}}$$ 由于 $p \mod 10$ 是完全平方数，可以得到 5 是模 p 意义下的二次剩余。可以令 $u = \frac{1+\sqrt{5}}{2}$，$v = \frac{1-\sqrt{5}}{2}$，$w = \frac{1}{\sqrt{5}}$，则 $f_n = w(u^n + v^n)$。可以得到 $u^n + v^n = cw^{-1}$。 由于 $v = -\frac{1}{u}$，可以得到 $u^n + (-\frac{1}{u})^n = cw^{-1}$。分 n 为奇数和偶数两种情况讨论，分别得到 $u^n + u^{-n} = cw^{-1}$ 和 $u^n - u^{-n} = cw^{-1}$ 两种情况。对于两种情况都可以解出 u^n。需要判断一个数是否是模 p 的二次剩余和求出模 p 的平方根。这些可以用经典算法解决。 得到 u^n 后转化为离散对数问题。可以用大步小步算法解决。		
时空复杂度	时间复杂度 $O(\sqrt{p})$，空间复杂度 $O(\sqrt{p})$。		

试题编号	DEG3MAXT	试题名称	Three-Degree-Bounded Maximum Cost Subtree
题目大意	给定一个带权无向图，保证每个点双连通分量的大小不超过 9。求出选出一些边和一些点使得这些点形成一棵树且每个点度数不超过 3 的最大总边权和达到最大总边权的方案数。		
算法讨论	可以考虑 DP，$f[i]$ 表示每个点度数状态为 i 的答案。对于每个至少有一条边的状态，由于所有点形成一棵树，一定可以找到一个度数为 1 的点。找到编号最小的度数为 1 的点，去掉这个点和连接的边，可以得到另一个状态，这样可以在 $O(n4^n)$ 的时间内得到答案。 由于每个点双连通分量的大小不超过 9，可以对每个双连通分量求出答案。由于每个点双连通分量的状态只与内部的连边和连接的割点的状态有关，可以 DP 得到答案。		
时空复杂度	时间复杂度 $O(m + n4^s)$，空间复杂度 $O(4^s)$，其中 s 为双连通分量的大小。		

试题编号	MONOPLOY	试题名称	Gangsters of Treeland
题目大意	有一棵树，一开始每个点有一个不同的权值。一个点的路径长度为这个点到根的路径上经过的点中权值变化的次数。有一些询问，每个询问会将根到一个点的路径上的所有点变为一个新的权值，或询问一棵子树内的所有点的路径长度的平均值。		
算法讨论	每个点的路径长度为这个点到根的路径上不同的权值个数减 1。可以维护 LCT，则同一个 Splay 树中的点有同样的权值。这样将根到一个点的路径上的所有点变为一个新的权值为 access 操作。对于每条链可以发现在这条链的最高结点的子树内的所有点的路径长度会增加 1。这样可以对每条链维护最高结点，在 access 操作时维护在 DFS 序的一个区间内增加或减少答案。询问时查询 DFS 序的一个区间的答案之和，得到平均值。		
时空复杂度	时间复杂度 $O(n\log^2 n)$，空间复杂度 $O(n)$。		

试题编号	QPOINT	试题名称	Queries With Points
题目大意	给定一些简单多边形，保证互不相交。给定一些询问，每次询问一个点在哪个多边形内部，或不在某个多边形内部。强制在线。		
算法讨论	如果没有强制在线，可以用扫描线并维护每个多边形的上边界和下边界的相对位置关系。将询问排序，每次询问时在平衡树中查询在哪两条边之间。 加上强制在线，可以用可持久化平衡树维护，这样询问时可以找到对应的平衡树，再查询答案。		
时空复杂度	时间复杂度 $O(s\log s + q\log s)$，空间复杂度 $O(s\log s)$，其中 s 为所有多边形的边数之和。		

试题编号	QTREE6	试题名称	Query on a tree VI
题目大意	给定一棵树，每个点是黑色或白色，初始时为黑色。有一些询问，每次询问与一个点有相同颜色的点中连通的点的个数，或改变一个点的颜色。		
算法讨论	可以用 LCT 维护相同颜色的点的连通性。每次询问相当于询问一个连通块的大小。但是修改时需要断开所有相连的边并加入一些边，当点的度数很大时会超时。可以对每个点维护两个特殊点，分别与这个点的孩子结点中的黑色点和白色点连边。这样修改一个点的颜色时需要断开这个点和一个特殊点的边并连接到另一个，同时断开与父亲结点的特殊点的边并连接到另一个特殊点。		
时空复杂度	时间复杂度 $O(m\log n)$，空间复杂度 $O(n)$。		

试题编号	REALSET	试题名称	Petya and Sequence
题目大意	给定一个长度为 n 的序列 a，问是否能找到长度为 n 的不全为 0 的序列 b，满足 $\sum_{i=0}^{n-1} a_i b_{(i+j) \mod n} = 0$ 对于所有 j 成立。		
算法讨论	构造矩阵 A，满足 $A_{ij} = a_{(i+j) \mod n}$，则相当于问 $Ax = 0$ 是否有非平凡解。这等价于矩阵 A 满秩。 构造多项式 $f(x) = \sum_{i=0}^{n-1} a_i x^i$,令 d 为 $\gcd(f(x), x^n-1)$ 的度数,则 A 的秩为 $n-d$。这样只需要判断是否 $d = 0$, 即 $\gcd(f(x), x^n-1)$ 为常数。这等价于 $f(x)$ 与 x^n-1 互质。 由于 $x^n - 1$ 分解因式后得到 $\prod_{d\mid n} \Phi_d(x)$，其中 $\Phi_d(x) = \prod_{\gcd(k,d)=1}(x-e^{\frac{2\pi ik}{n}})$。只需要判断是否对每个 d 满足 $\Phi_d(x) \nmid f(x)$。 令 d 的所有质因子为 $p_1, p_2, \dots, p_m$。注意到 $\Phi_d(x) \mid f(x)$ 当且仅当 $x^d - 1 \mid f(x)\prod_{i=1}^m(x^{\frac{d}{p_i}} - 1)$，因为 $\frac{x^d-1}{\Phi_d(x)}$ 中的每个部分可以在 $x^{\frac{d}{p_i}}-1$ 中得到。这样只需要在模 x^d-1 意义下计算 $f(x)\prod_{i=1}^m(x^{\frac{d}{p_i}} - 1)$。 由于模 x^d-1 和乘 $x^{\frac{d}{p_i}}-1$ 的计算容易在 $O(d)$ 时间内完成，可以在 $O(n + ds(d))$ 的时间内对一个 d 判断，其中 $s(d)$ 为 d 的质因子个数。这样可以在 $O(n\tau(n) + \sigma(n)\log n)$ 的时间内得到答案。		
时空复杂度	时间复杂度 $O(n\tau(n) + \sigma(n)\log n)$，空间复杂度 $O(n)$。		

试题编号	CNTDSETS	试题名称	Counting D-sets
题目大意	定义两个 n 维点之间的距离为对应维的坐标之差的最大值。问有多少不等价的点集，使得点集中每个点坐标为整数且点对距离的最大值为 d。两个点集等价，当且仅当可以将每个点平移同一个向量后变为另一个点集。		
算法讨论	可以求出最大距离不超过 d 的点集个数，再减去最大距离不超过 $d-1$ 的点集个数。考虑求最大距离不超过 d 的点集个数。由于要求不等价，可以限制所有坐标的最小值为 0。这样要求至少有一个坐标有 0。可以求出有 k 维坐标不能取 0 的方案数，再容斥得到答案。这样可以在范围内选择任意一个子集。可以求出范围内的点数 s，则答案为 2^s。		
时空复杂度	时间复杂度 $O(tn\log m)$，空间复杂度 $O(n)$，其中 m 为模数。		

试题编号	TAPAIR	试题名称	Counting The Important Pairs
题目大意	给定一个简单无向连通图，问有多少种方案删除两条边使得这个图仍然连通。		
算法讨论	求出这个图的 DFS 树。注意到删除树边后需要一条覆盖这条边的非树边使得树连通。给每条非树边一个不同的编号，在每条树边上记录能够覆盖这条边的所有非树边的编号。如果删除两条非树边，则没有限制。如果删除一条非树边和一条树边，则当这条树边只被一条非树边覆盖时不能删除这条非树边。如果删除两条树边，则不能删除编号集合相同的树边，否则中间一部分不会连通。这样要求不能删除编号集合相同的边。 对每条非树边记录一个随机数，每条树边记录所有能覆盖这条边的非树边的数的异或。这样可以用记录的数相等判断编号集合相等。可以证明错误概率很小。求出这个数时可以对每条非树边的两个端点上异或这条非树边的随机数，每条树边的数为子树内所有数的异或。		
时空复杂度	时间复杂度 $O(m \log m)$，空间复杂度 $O(m)$。		

试题编号	DAGCH	试题名称	Graph Challenge
题目大意	给定一个 n 个点的有向图，保证这个图的 DFS 序为 1 到 n。对每个点 u ，得到编号最小的 v 满足有一条从 v 到 u 且中间所有点的编号大于 u 的路径。有一些询问，每次询问有多少点的对应点为给定的点。		
算法讨论	显然只需要找出每个点的对应点。首先可以在图中找到 DFS 序上每个点的父亲结点，为编号最小的向这个点连边的点。对于每条从编号小的点连向编号大的点的边，可以直接用来更新答案。对于每条从编号大的点连向编号小的点的边，相当于用只经过编号大于 u 的点能够到达 v 的点的最小答案来更新答案。 这样可以按照编号从大到小加入点，可以用并查集维护能够到达一个点的所有点中答案的最小值。		
时空复杂度	时间复杂度 $O(m\alpha(n))$，空间复杂度 $O(n)$。		

试题编号	COT5	试题名称	Count on a Treap
题目大意	你需要维护一棵 Treap，要求支持插入一个元素，删除一个元素，询问两个元素在树上的距离。		
算法讨论	首先将所有权值离散化。可以在线段树上维护当前的权值。对于每个询问，可以先找出两个点的 LCA，再求出点的深度。两个点的 LCA 为线段树上区间最大值的位置。对于每个点的深度，可以分为一个点是父亲的左孩子或右孩子两种情况。相当于权值比这个点小或大的点。在一个点左边第一个比这个点大的点为这个点的父亲，右边同理。这样需要在线段树上维护前缀中每次找到一个点的父亲的次数。 每次询问时，可以分为线段树上的一些区间得到答案。在线段树上的每个区间中，如果当前值比右边的最大值大，则只需要考虑左子树，否则需要考虑右子树和在已经为右边最大值的情况下左边的答案。这样需要维护右边的最大值在左边产生的答案。可以在修改时维护。		
时空复杂度	时间复杂度 $O(n\log^2 n)$，空间复杂度 $O(n)$。		

试题编号	GERALD07	试题名称	Chef and Graph Queries
题目大意	给定一个无向图，有一些询问，每次询问只考虑编号在一个区间内的边时图中有多少个连通块。		
算法讨论	将询问按照右端点排序，每次加入一条边，则每个询问为询问当前已经加入的边中编号不小于一定值的边形成的连通块个数。 维护当前已经加入的边形成的最大生成森林，则只需要考虑这些边，因为剩下的边一定不会连接两个连通块。每次询问只需要求出在这些边中有多少边的编号不小于一定值。可以用 LCT 维护最大生成森林，用树状数组维护边的编号。		
时空复杂度	时间复杂度 $O(n+m\log m+q\log m)$，空间复杂度 $O(n+m+q)$ 。		

试题编号	STREETTA	试题名称	The Street
题目大意	有 n 个位置，每个位置有两个权值。有一些操作，第一种操作将一个区间内的位置的第一个权值和一个等差数列的值取较大值，另一种操作将一个区间内的位置的第二个权值加上一个等差数列的值。每次询问一个位置的两个权值之和。		
算法讨论	每一个等差数列的权值可以看作对第 i 个位置有一个 $ai+b$ 的权值。对于第二种操作容易在线段树上维护。这样只需要考虑第一类操作。 每个第一类操作在线段树上对应一些区间。对于一个区间，如果已经有一个等差数列的权值，则新加入的等差数列有两种情况。如果在这个区间内一个权值一定比另一个大，则可以将权值设定为较大的权值。否则由于两个权值都是线性的，一定可以找到一个分界点，使得两边分别为一个权值。 这样在线段树的两个子树中一定有一边为同一个权值。这样可以将这个子树的权值变为较大的权值，再递归在另一个子树中更新权值。这样可以做到 $O(m\log^2 m)$。		
时空复杂度	时间复杂度 $O(m\log^2 m)$，空间复杂度 $O(m)$。		

试题编号	GERALD08	试题名称	Chef and Tree Game
题目大意	给定一棵树，每条边为黑色或白色。两个人轮流操作，第一个人可以删除一条和根连通的黑色边，第二个人可以删除一条和根连通的白色边。不能操作的人输。问在两个人分别为先手时谁有必胜策略。		
算法讨论	对每棵子树求出一个值 $f(u)$，当 $f(u)=0$ 时后手胜，当 $f(u)>0$ 时第一个人胜，当 $f(u)<0$ 时第二个人胜。对于 $f(u)$，求出所有 u 的子树对 $f(u)$ 的贡献。设这棵子树的值为 x。当这条边为黑色时贡献为 $\frac{x+p}{2^{p-1}}$，其中 p 为最小的使得 $x+p>1$ 的整数 p。当这条边为白色时贡献为 $\frac{x-p}{2^{p-1}}$，其中 p 为最小的使得 $x-p<-1$ 的整数 p。$f(u)$ 为所有子树的贡献之和。 对于每棵子树的 $f(u)$，可以分开存储 $f(u)$ 的整数部分和小数部分。这样对于每个 x，容易快速求出 p。对于小数部分可以用 set 存储为 1 的所有位。进行右移操作时可以对整个小数部分打标记，这样右移 p 位可以在 $O(p\log n)$ 的时间内完成。加法时可以对于小数部分的每一位合并，用启发式合并的方法选择较小的合并到较大的。 由于最多只会增加 $O(n)$ 个 bit，而每一位只会被减少一次，因此总时间复杂度为 $O(n\log^2 n)$。		
时空复杂度	时间复杂度 $O(n\log^2 n)$，空间复杂度 $O(n)$。		

试题编号	ANUDTQ	试题名称	Dynamic Trees and Queries
题目大意	给定一棵树，每个点有一个权值。有一些操作，每个操作可以增加一个点作为一个点的孩子，可以对一棵子树内的权值增加一个值，可以删除一棵子树，可以询问一棵子树内的权值和。强制在线。		
算法讨论	注意到只有对子树的询问，可以维护树的 DFS 序。增加一个点相当于在 DFS 序中插入一个元素，子树修改相当于在 DFS 序中修改一段区间，删除子树相当于在 DFS 序中删除一段，子树询问相当于询问 DFS 序中一段区间的权值和。可以用平衡树维护 DFS 序。		
时空复杂度	时间复杂度 $O(n\log n)$，空间复杂度 $O(n)$。		

试题编号	SEINC	试题名称	Sereja and Subsegment Increasings
题目大意	有两个长度为 n 的序列 a 和 b，每次可以将 a 的一个区间内的数加 1 。问最少用多少次可以将 a 和 b 变为模 4 意义下相等。		
算法讨论	显然只需要考虑将 a 减少 1，变为全部为 0 的情况。如果不考虑模 4 的情况，贪心可以得到答案为 $\sum_{i=1}^{n-1}\max(a_{i+1}-a_i,0)$ 如果考虑模 4 的情况，相当于可以对 a 中的一些位置加上 4 的倍数，使得答案最小。 可以构造序列 $c_i=a_{i+1}-a_i$，则可以选择一些 $i<j$，将 c_i 增加 4，将 c_j 减少 4，使得 $\sum_{i=1}^{n-1}\max(c_i,0)$ 最小。 显然一个位置最多只会增加或减少一次。只有当一个位置大于 0 时会减少这个位置，并且之前增加的位置一定小于 0。可以记录之前为-3 和-2 的位置个数，这样可以考虑贪心选择最优方案，增加最小的数。		
时空复杂度	时间复杂度 $O(n)$，空间复杂度 $O(n)$。		

试题编号	TWOCOMP	试题名称	Two Companies
题目大意	给定一棵树和一些路径，路径分为两类。每条路径有一个权值。你需要选择一些路径，使得选择的不同类型的路径没有公共点。问能够选出的最大总权值。		
算法讨论	首先求出任意两条路径是否有公共点。可以预处理 LCA 再用 $O(\log n)$ 时间判断每一对路径是否有公共点。这样可以在 $O(n\log n+m^2\log n)$ 的时间内求出任意两条路径是否有公共点。接下来问题变为二分图最大权独立集问题，可以用最大流解决。		
时空复杂度	时间复杂度 $O(n\log n+m^2\log n+m^4)$，空间复杂度 $O(n+m^2)$。		

试题编号	SEAARC	试题名称	Sereja and Arcs
题目大意	一条直线上有 n 个点，每个点有一个颜色。在任意两个颜色相同的点之间连接一个半圆，问有多少对半圆相交且对应颜色不同。		
算法讨论	对出现次数大于 s 和小于 s 的颜色分开考虑。对于出现次数大于 s 的颜色，可以枚举每种颜色，求出其它颜色的半圆与这个颜色的半圆相交的次数。对于另一种颜色的每个半圆，可以得到答案为 $b(a+c)$，其中 a, b, c 分别为这个半圆将直线分为的三个部分中枚举的颜色的点的个数。这样可以枚举另一种颜色的一个点，统计这个点与之前点的半圆对应的答案。注意到只需要维护所有 a 的和与平方和，可以在 $O(n)$ 时间内得到枚举这种颜色的答案。这一部分的时间复杂度为 $O(\frac{n^2}{s})$。 对于所有出现次数小于 s 的颜色，可以按照半圆的右端点从左到右加入。对于每个点，找到与之前所有点形成的半圆，则所有覆盖左端点的半圆与这个半圆相交。可以用树状数组维护覆盖每个点的半圆个数。这一部分的时间复杂度为 $O(ns\log n)$。 取 $s=\sqrt{\frac{n}{\log n}}$ 时达到最优，时间复杂度为 $O(n\sqrt{n\log n})$。		
时空复杂度	时间复杂度 $O(n\sqrt{n\log n})$，空间复杂度 $O(n)$。		

试题编号	GNUM	试题名称	Game of Numbers
题目大意	给定两个序列 a 和 b。你需要进行一些操作，每次选择两对整数 (i,j) 和 (p,q)，使得 $a_i<b_j$，$a_p>b_q$，$\gcd(a_i,a_p,b_j,b_q)>1$，且 (i,j) 和 (p,q) 之前没有被选择过。问最多能进行多少次操作。		
算法讨论	对于每一对满足 $a_i<b_j$ 且 $\gcd(a_i,b_j)>1$ 的 (i,j)，可以建出一个点，类似对于每一对满足 $a_p>b_q$ 且 $\gcd(a_p,b_q)>1$ 的 (p,q)，可以建出一个点。如果 (i,j) 和 (p,q) 满足 $\gcd(a_i,a_p,b_j,b_q)>1$，则在对应的点之间连边。这样可以用最大流算法解决。 但是这样构造出的图边数太多。可以从 (i,j) 向 $\gcd(a_i,b_j)$ 的每个质因子连边，从 $\gcd(a_p,b_q)$ 的每个质因子向 (p,q) 连边。这样只需要对这些数分解质因数。可以对 a 和 b 的每个数分解质因数，则可以得到所有 $\gcd(a_i,b_j)$ 的质因子。		
时空复杂度	时间复杂度 $O(n\sqrt{s}+n^3\log^{1.5}s)$，空间复杂度 $O(n^2\log s)$，其中 s 为最大的数。		

试题编号	SEAEQ	试题名称	Sereja and Equality
题目大意	定义两个序列 a 和 b 相似，当且仅当长度相等且对于每个 i，在 a 中比 a_i 小的数的个数等于在 b 中比 b_i 小的数的个数。给定 n，问在所有的 n 个元素的排列 p，q 中，这两个排列的相似且不超过 e 个逆序对的区间个数之和。		
算法讨论	两个排列相似当且仅当相对大小关系相同。对于每个长度的区间，可以计算出需要统计的次数。对于每一种长度为 k 的区间，可以在长度为 n 的序列中有 $n-k+1$ 个位置，且可以有 $\frac{n!}{k!}$ 种方案选择剩下的数。这样需要求出长度为 k 的逆序对个数不超过 e 的序列个数。 可以考虑在长度为 $k-1$ 的区间内插入一个数，有 k 种方案，分别增加 0 到 $k-1$ 个逆序对。这样可以用 $O(n^3)$ 的 DP 求出方案数。		
时空复杂度	时间复杂度 $O(n^3+tn)$，空间复杂度 $O(n^3)$。		

试题编号	QRECT	试题名称	Rectangle Query
题目大意	有一些操作，每次会加入一个边与坐标轴平行的矩形，删除一个矩形，或询问有多少矩形与一个边与坐标轴平行的矩形有公共点。		
算法讨论	首先预处理出每个删除操作对应的矩形，将删除操作看作加入一个权值为 -1 的矩形。两个矩形有公共点当且仅当两个矩形在某一维上的投影有公共点。可以考虑容斥，求出在 x 上没有公共点，在 y 上没有公共点，在 x 和 y 上都没有公共点的答案。 在 x 上没有公共点的答案可以分为在右边和左边两种情况。这样需要考虑一个矩形的左边界在另一个矩形的右边界右边的情况。可以用树状数组维护。在 y 上没有公共点的情况类似。 在 x 和 y 上都没有公共点的答案可以按照方向分为四种情况。可以分治解决。每次考虑左边的所有矩形对右边的所有询问的贡献。可以按照 x 从小到大加入，按照 y 维护树状数组。		
时空复杂度	时间复杂度 $O(n\log^2 n)$，空间复杂度 $O(n)$。		

试题编号	FIBTREE	试题名称	Fibonacci Numbers on Tree
题目大意	给定一棵树，每个点有一个权值，初始时为 0。有一些询问，每次可以改变一条路径上的权值，对路径上第 i 个点增加 f_i 的权值，其中 f_i 为斐波那契数。可以询问以某个点为根时一棵子树的权值和，询问一条路径上的权值和，将当前状态变为第 k 次询问之后的状态。强制在线。		
算法讨论	对树进行树链剖分，则一条路径变为一些区间，一个子树变为一个区间。求出每个区间内第一个点是路径上的第几个点和区间内的方向。对于每个区间记录 a，b，c，d 表示这个区间中的每个位置 i 要增加 $af_i + bf_{i+1} + cf_{-i} + df_{-i+1}$ 的权值。 对于每个区间，由于 $f_{x+i} = f_{x-1}f_i + f_xf_{i+1}$，可以将修改权值变为这些标记。对于每个询问，由于 $\sum_{i=1}^{n} f_i = f_{n+2} - 1$，可以在区间内求和，剩下三个情况的求和类似。由于有恢复状态的操作，需要用可持久化线段树维护权值。		
时空复杂度	时间复杂度 $O(n + m\log^2 n)$，空间复杂度 $O(m\log^2 n)$。		

试题编号	TRIPS	试题名称	Children Trips
题目大意	给定一棵树，每条边有一个权值，权值为 1 或 2。有一些询问，每次询问从一个点到另一个点，每一步走总边权不超过 p 的路径，问需要多少次能够走到另一个点。		
算法讨论	将树按照 s 的深度分块。对于每个点求出这个点往上走到的第一个深度为 s 的倍数的点和这一段路径的长度。预处理出从每个点开始往上走，总边权不超过一定值，可以走到哪个点。这可以在 $O(ns)$ 时间内求出。再求出每次走总边权不超过一定值，多少次可以走到对应的深度为 s 的点以及最后一次剩下的权值。同样可以在 $O(ns)$ 时间内求出。 每次询问时，如果当前两个点对应的深度为 s 的倍数的点不同，则可以让深度较大的点走到对应的点。如果权值限制较小，则可以用预处理的需要的次数和最后剩下的权值，否则可以用路径长度。当权值限制不足时可以用预处理的值得到走到哪个点。当两个点在同一块内时可以暴力贪心得到答案。每次询问时间为 $O(s + \frac{n}{s})$。 当 $s = \sqrt{n}$ 时达到最优，时间复杂度为 $O((n+m)\sqrt{n})$。		
时空复杂度	时间复杂度 $O((n+m)\sqrt{n})$，空间复杂度 $O(n\sqrt{n})$。		

试题编号	BTREE	试题名称	Union on Tree
题目大意	给定一棵树。有一些询问，每个询问选择一些点，每个点有一个参数。问有多少个点距离至少一个选择的点的距离不超过这个点的参数。		
算法讨论	对于每次询问，可以建出虚树，则虚树上有一些点有一些参数。先对于每个点，用这个点的参数更新父亲结点的参数，再用父亲结点的参数更新这个点的参数。这样每个点的影响只会在相邻的边上。 考虑虚树上的每条边。可以找到这条边上的一个分割点，在分割点之上的点由父亲结点覆盖，在分割线之下的点由孩子结点覆盖。对于父亲结点覆盖的部分，需要求出一棵子树内深度不超过一定值的结点个数。对于孩子结点覆盖的部分，可以求出距离这个点不超过一定值的结点数减去在范围外的个数。对于在范围外的个数，可以相当于在范围外的距离分割点不超过一定值的结点个数。可以转化为距离分割点不超过一定值的结点个数减去子树内深度不超过一定值的结点个数。这样还需要求出距离一个点不超过一定值的结点个数。 可以对树进行点分治，则可以在 $O(\log n)$ 的时间内求出距离点不超过一定值的结点个数。子树内深度不超过一定值的点个数可以用可持久化线段树实现。		
时空复杂度	时间复杂度 $O(n\log n+s\log n)$，空间复杂度 $O(n\log n+s)$ ，其中 s 为询问的总点数。		

试题编号	FNCS	试题名称	Chef and Churu
题目大意	给定一个长度为 n 的序列和 n 个询问，每个询问为询问一个区间的数的和。有一些操作，每次修改一个数或询问一个区间内的询问的答案之和。		
算法讨论	将询问每 s 个分为一块。处理每一块时先求出序列的前缀和，再求出所有询问的答案，得到询问答案的前缀和。得到在这一块中询问的区间，则只需要处理出每一段中的答案，每次询问时可以用 $O(s)$ 的时间得到答案。每次修改会对每一段的答案有影响，可以处理出每一段的所有询问覆盖每个在这一块中被修改的点的次数，这样每一次修改可以在 $O(s)$ 的时间内完成。 需要求出每一段中的所有询问覆盖每个点的次数。由于每一段覆盖的点一定是一些区间，可以在 $O(n)$ 时间内求出每一段的覆盖区间，在 $O(q)$ 时间内求出每个点被覆盖的次数。 当 $s=\sqrt{q}$ 时达到最优，时间复杂度为 $O(n\sqrt{q}+q\sqrt{q})$。		
时空复杂度	时间复杂度 $O(n\sqrt{q}+q\sqrt{q})$，空间复杂度 $O(n+q)$。		

试题编号	SEAORD	试题名称	Sereja and Order
题目大意	有 n 个任务，每个任务要在两个位置分别执行，需要 a_i 和 b_i 的时间。一个任务不能在两个位置同时执行，一个位置不能同时执行两个任务。问执行所有任务的最少时间。		
算法讨论	显然答案至少为 $\max(\sum_i a_i, \sum_i b_i, \max_i(a_i+b_i))$。可以构造出一种解达到这个下界。 对于所有 $a_i > b_i$ 的任务，可以选出 a_i 最大的任务，让两个位置依次执行这个任务，先在第二个位置执行，再依次执行剩下的任务。对于 $a_i \leq b_i$ 的任务可以类似做。将一部分任务全部反向，将两部分任务拼接在一起，再将一个位置中执行的最后一个或第一个任务移动到最开头或最后，可以发现答案可以达到下界。		
时空复杂度	时间复杂度 $O(n)$，空间复杂度 $O(n)$。		

试题编号	DIVIDEN	试题名称	Divide or die
题目大意	给定一个 n 度角，其中 n 为整数，问是否能将这个角 n 等分，并输出方案。		
算法讨论	当 n 为 3 的倍数时，由于 60 度角不能三等分，可以得到 3 度角不能三等分，因此 n 度角不能 n 等分。 当 n 不是 3 的倍数时，可以构造出 3 度角，由于 $\gcd(n,3)=1$，一定可以得到 1 度角，因此可以 n 等分。 构造出 3 度角可以先构造出 60 度角和 36 度角，再将两个角的差平分 3 次得到。		
时空复杂度	时间复杂度 $O(n)$，空间复杂度 $O(1)$。		

试题编号	RIN	试题名称	Course Selection
题目大意	你有 m 个学期学习 n 门课。有一些课必须在另一些课之前学。在某个学期学习某门课会得到一个得分。你需要求出 n 门课的最高平均得分。		
算法讨论	需要求出最高总分，用最小割解决。对每一门课建出一条 m 个点的链，割掉每条边表示在这个学期学习这门课，流量为满分减去得分。如果一门课必须在另一门课之前学，则对于它的每一个点，向另一门课的下一学期的点连边，流量为无穷大，表示不能这门课在 i 之后而另一门课在 i 之前。求出最小割后得到答案。		
时空复杂度	时间复杂度 $O(n^2(k+n)m^3)$，空间复杂度 $O((n+k)m)$。		

<table>
<tr><td>试题编号</td><td>RANKA</td><td>试题名称</td><td>Ranka</td></tr>
<tr><td>题目大意</td><td colspan="3">构造出在 9×9 棋盘上从空白状态开始 n 步的围棋对局。要求不能出现重复状态。$n = 10000$。</td></tr>
<tr><td>算法讨论</td><td colspan="3">考虑这个构造：
.ox.
o.ox
.ox.
这样可以不断在两个状态之间重复。
考虑这个构造：
x.xox.x.x
.xo.ox.xo
x.xox.x..
.xo.ox.xo
x.xox.x.x
.xo.ox.xo
x.xox.x..
.xo.ox.xo
x.xox.x.x
在这个构造中共有 13 个可以重复状态的构造。用格雷码遍历所有的状态，每次修改一个位置。这样一共可以得到 2^{13} 个状态。由于在状态之间转移时有一半情况需要一步，另一半情况需要两步，可以得到大约 1.5×2^{13} 步，可以通过本题。</td></tr>
<tr><td>时空复杂度</td><td colspan="3">时间复杂度 $O(n)$，空间复杂度 $O(1)$。</td></tr>
</table>

<table>
<tr><td>试题编号</td><td>XRQRS</td><td>试题名称</td><td>Xor Queries</td></tr>
<tr><td>题目大意</td><td colspan="3">有一个初始时为空的序列。有一些操作，每次在序列最后加入一个数，删除最后 k 个数，询问一个区间内与给定的数异或最大的数，询问一个区间内不超过给定的数的数的个数，询问一个区间内的第 k 小数。</td></tr>
<tr><td>算法讨论</td><td colspan="3">注意到这些询问都可以用可持久化线段树维护。询问一个区间内与给定的数异或最大的数可以贪心找到使每一位最大的数。询问一个区间内不超过给定的数的数的个数可以在线段树上查询。询问一个区间内的第 k 小数可以在线段树上二分。
插入操作可以直接修改可持久化线段树得到。删除操作只要删除最后的一些线段树。</td></tr>
<tr><td>时空复杂度</td><td colspan="3">时间复杂度 $O(m \log m)$，空间复杂度 $O(m \log m)$。</td></tr>
</table>

试题编号	TREECNT2	试题名称	Counting on a Tree
题目大意	给定一棵树，每条边有一个权值。有一些修改操作，每次修改一条边的边权。求出在最开始和每次修改后，有多少对点之间的路径上边权的 gcd 为 1 。		
算法讨论	显然只需要求出路径上边权都为 d 的方案数 $f(d)$，则答案为 $\sum_{d\geq 1}\mu(d)f(d)$。这个方案数可以用只考虑权值为 d 的倍数的边形成的连通块大小得到。需要考虑的总边数为所有边权的约数个数和，不会很多。 当有修改时，可以特殊考虑被修改过的边。每条边可以在每个询问状态中有不同的权值，因此对于每个询问状态的贡献分开考虑。这样总共会有 $O(q^2)$ 条边。 注意求出一条边会在哪些 d 被考虑时需要分解质因数，可以预处理最小质因子加速。		
时空复杂度	时间复杂度 $O(l+(n+q^2)s\alpha(n))$，空间复杂度 $O(l+(n+q^2)s)$。		

试题编号	RNG	试题名称	Random Number Generator
题目大意	给定一个 k 阶常系数齐次线性递推关系，和这个递推关系的前 k 项，求出第 n 项。		
算法讨论	由于 $a_n-\sum_{i=1}^{k}c_ia_{n-i}=0$，可以构造出多项式 $f(x)=x^k-\sum_{i=1}^{k}c_ix^{k-i}$，则求出 $x^n \bmod f(x)$ 的系数就可以求出第 n 项与前 k 项的关系。 要求 $x^n \bmod f(x)$，可以先求出 $g(x)=x^{\lfloor\frac{n}{2}\rfloor} \bmod f(x)$，则只需要求出 $g^2(x) \bmod f(x)$ 或 $g^2(x)x \bmod f(x)$。求出 $g^2(x)$ 可以用 FFT 在 $O(k\log k)$ 时间内求出。对 $f(x)$ 取模可以用多项式除法在 $O(k\log k)$ 时间内求出。这样可以在 $O(k\log k\log n)$ 时间内得到答案。		
时空复杂度	时间复杂度 $O(k\log k\log n)$，空间复杂度 $O(k)$。		

试题编号	BWGAME	试题名称	Black-white Board Game
题目大意	给定一个 $n \times n$ 的矩阵，每一行中有一个区间为黑色。问在所有满足对于所有 i 满足矩阵中第 i 行第 p_i 列为黑色的排列 p 中，逆序对为奇数和偶数的排列哪个多。		
算法讨论	将黑色看作 1，则只需要求出矩阵的行列式的符号。可以用高斯消元解决。每次选出某一列为 1 的行，并将剩下所有这一列为 1 的行消去。 注意到如果每次选择区间最短的行，则需要消去的行消去后仍然是一个区间中为 1。这些区间的右端点不变。这样可以对每个位置维护所有左端点为这个点的区间的右端点，每次需要找到右端点最小的行，再将剩下所有区间的左端点变为选择行的右端点。如果不能找到这一列为 1 的行，则行列式为 0。 这样可以用可并堆维护。最后只需要求出交换行的次数，就能得到答案。		
时空复杂度	时间复杂度 $O(n \log n)$，空间复杂度 $O(n)$。		

试题编号	LPARTY	试题名称	Little Party
题目大意	给定一些长度为 n 的 01 串。一个选择为一个长度为 n 的串，有一些位置确定为 0 或 1，另一些位置不确定，这个选择的权值为确定为 0 或 1 的个数。一个选择能覆盖所有确定的位置相同的 01 串。你需要找出一些选择，使得这些选择覆盖的所有串的并集为给定的所有串。求最小总权值。		
算法讨论	称一个选择为合法的，如果它覆盖的所有串都在输入中。显然如果一个合法选择覆盖的集合为另一个合法选择覆盖的集合的子集，则这个选择可以忽略。可以按照权值从小到大枚举合法选择，如果覆盖的集合不是之前任意一个合法选择覆盖的集合的子集，则加入这个选择。可以证明这样得到的合法选择最多只有 $O(2^n)$ 个。 可以搜索选择哪些合法选择，这样时间复杂度为 $O(2^{2^n})$。可以加上一些剪枝。如果选择剩下所有的合法选择仍然不能覆盖输入的所有串，则直接退出。如果当前的总权值大于答案，则直接退出。可以按照权值从小到大搜索，每次优先搜索使用这个合法选择的情况。这样可以通过本题。		
时空复杂度	时间复杂度 $O(2^{2^n})$，空间复杂度 $O(2^n)$。		

<table>
<tr><td>试题编号</td><td>CBAL</td><td>试题名称</td><td>Chef and Balanced Strings</td></tr>
<tr><td>题目大意</td><td colspan="3">给定一个字符串。有一些询问，每次询问一个子串中所有满足每个字符出现次数为偶数的子串的长度的 k 次方和。$k \leq 2$。强制在线。</td></tr>
<tr><td>算法讨论</td><td colspan="3">先求出每个前缀中每个字符出现次数的奇偶性。这样一个子串满足条件当且仅当两个端点的值相等。这样只需要找到一个区间内所有相同的数。
将每 s 个数分为一块，求出以每个块边界为左端点或右端点的区间内的答案。可以固定一个端点，另一个端点扫过剩下的位置，维护每个数的所有出现位置的个数，和，平方和。这样可以在 $O(n)$ 的时间内对一个端点预处理。这一部分的时间复杂度为 $O(\frac{n^2}{s})$。
对于每个询问，只需要求出一个数在左端点所在块，一个数在右端点所在块的情况，剩下的情况可以用预处理的答案求出。这一部分可以用类似的方法求出，时间复杂度为 $O(s)$。
当 $s = \sqrt{n}$ 时达到最优，时间复杂度为 $O(n\sqrt{n})$。</td></tr>
<tr><td>时空复杂度</td><td colspan="3">时间复杂度 $O(n\sqrt{n})$，空间复杂度 $O(n\sqrt{n})$。</td></tr>
</table>

试题编号	CLOWAY	试题名称	Future of draughts
题目大意	给定 t 个无向图。一开始在每个图中选择一个点，之后每次在至少一个图中将点移动到一个相邻的位置。有一些询问，每次询问只考虑一个区间内的图的情况下有多少种方案在 k 步以内将所有点移动到一开始的位置。答案模 10^9+7。		
算法讨论	对于每个图，考虑它的邻接矩阵 A。显然在 k 次后移动到一开始的位置的方案数为 A^k 的迹。如果 A 的特征值为 $\lambda_1, \lambda_2, \dots, \lambda_n$，则 A^k 的特征值为 $\lambda_1^k, \lambda_2^k, \dots, \lambda_n^k$。$A^k$ 的迹为 $\sum_{i=1}^n \lambda_1^k$。 考虑有两个图的情况。可以构造一个新图，其中每个点对应之前两个图的一对点。连边时需要在至少一个图中连通。可以证明如果两个图的特征值分别为 $\lambda_{a1}, \lambda_{a2}, \dots, \lambda_{an}$ 和 $\lambda_{b1}, \lambda_{b2}, \dots, \lambda_{bm}$，则新图的特征值为 $\lambda_{ai}\lambda_{bj}+\lambda_{ai}+\lambda_{bj}$，对于所有 i 和 j。这样在有 t 个图的情况中特征值为 $\prod_{i=1}^t(\lambda_{ik_i}+1)-1$。 这样需要求出 $(\prod_{i=1}^t(\lambda_{ik_i}+1)-1)^k$ 的和。展开得到 $\sum_{p=0}^k(-1)^{k-p}\binom{k}{p}\prod_{i=1}^t(\lambda_{ik_i}+1)^k$。只需要求出 $\prod_{i=1}^t(\lambda_{ik_i}+1)^k$。 注意到 λ_i+1 是 $A+I$ 的特征值。令 $B=A+I, f(x)=\sum_{i=0}^n c_i x^i$ 为 B 的特征多项式，则 $f(B)=0$。这样得到 $B^k=\sum_{i=1}^n -c_{k-i}B^{k-i}$，即 $\operatorname{tr}(B^k)=\sum_{i=1}^n -c_{k-i}\operatorname{tr}(B^{k-i})$。由于容易求出这个递推的前 n 项，可以求出 $\operatorname{tr}(B^k)$。这样还需要求出 B 的特征多项式。 注意到 $c_i=(-1)^{n-i}\sum_{p_1<p_2<\dots<p_{n-i}}\prod_{j=1}^{n-i}\lambda_{p_j}$。这个式子可以用容斥求出。 在求出 $\sum_{p=0}^k(-1)^{k-p}\binom{k}{p}\prod_{i=1}^t(\lambda_{ik_i}+1)^k$ 时，直接暴力是 $O(k^2)$ 的，不能通过。注意到这是一个卷积，可以用 FFT 求出。由于模数不适合 NTT，可以选择三个不同的模数进行 NTT，用中国剩余定理得到不取模的答案，再取模得到答案。 可以对每个区间预处理答案，在 $O(1)$ 时间回答每个询问。		
时空复杂度	时间复杂度 $O(tn^4+tnk+t^2k\log k+q)$，空间复杂度 $O(t^2k)$。		

试题编号	DISTNUM	试题名称	Simple Queries
题目大意	给定一个序列。有一些询问，每个询问会询问 $\sum_{1\le i<j<k\le l} s_i s_j s_k$，其中 $s_1, s_2, \dots, s_l$ 为一个区间内的所有不同的数，修改一个数，删除一个数，插入一个数，询问区间内的不同数的个数。		
算法讨论	首先处理所有插入和删除操作，删除时只打上删除标记，得到最后的序列。这样变为只有修改操作。只需要维护区间内不同数的 k 次方和，其中 $0\le k\le 3$，所有询问的答案可以计算得到。 对每个数记录之前最近一个相同的数 p_i，则一个区间 $[l,r]$ 中不同的数为满足 $l\le i\le r$ 且 $p_i<l$ 的数。可以用二维数据结构维护。对于每个修改操作，只会有常数个位置的 p 会改变。可以对每个值维护一个 set 表示出现的位置，则可以快速找到这些位置。		
时空复杂度	时间复杂度 $O((n+q)\log^2 n)$，空间复杂度 $O((n+q)\log^2 n)$ 。		

2 Challenge 试题

试题编号	STEPAVG	**试题名称**	Stepping Average
题目大意	给定一些整数，每次可以将两个数删除并加入它们的平均值，直到最后剩下一个数。要求剩下的数尽可能接近给定的整数 k。		
算法讨论	考虑用贪心算法。每次找到当前最小和最大数的平均值。如果需要的平均值与这个数接近，则删除最小和最大的数。如果需要的平均值较小，则可以令最后一步为将最小的数和之前的值合并。可以删除最小的数并计算出剩下的数中应该接近的数。需要的平均值较大时同理。在一定范围内尝试不同的参数可以通过本题。		
时空复杂度	时间复杂度 $O(tn^2)$，空间复杂度 $O(n)$，其中 t 为尝试的次数。		

试题编号	SIMGRAPH	**试题名称**	Similar Graphs
题目大意	给定两个点数相同的无向图，你需要对两个图的结点重新标号，使得在两个图中的公共边尽可能多。		
算法讨论	首先随机得到一种标号。接下来模拟退火，每次尝试交换两个点的标号。注意到每次只改变了两个点的标号，可以在 $O(n)$ 的时间内计算出新的答案。		
时空复杂度	时间复杂度 $O(n^2+tn)$，空间复杂度 $O(n^2)$，其中 t 为模拟退火的步数。		

试题编号	CKROACH	**试题名称**	Killing Gs
题目大意	有一些物品，每个物品有 n 个参数 a_{ij}。选择每个物品有一个费用。你需要选择一些物品，使得对于所有 j 满足 $\prod(1-a_{ij}) \geq 0.9$。要求费用尽可能小。		
算法讨论	显然可以取对数，变为对于每个参数，选择的物品的这些参数之和至少为 1。考虑一种贪心算法，每次选择效率最高的物品，直到满足条件。可以用所有没有达到 1 的参数的和除以费用来判断效率。 用贪心得到一组解后可以用模拟退火，每次尝试删除一个物品并加入另一个物品。每次尝试可以在 $O(n)$ 时间内判断是否合法。		
时空复杂度	时间复杂度 $O(m^2\log m+m^2n+tn)$，空间复杂度 $O(m+n)$，其中 t 为模拟退火的步数。		

试题编号	SIMNIM	试题名称	Simultaneous Nim
题目大意	给定 n 个数，保证所有数的异或为 0。要求把这些数分为尽可能多的部分，使得每一部分的数的异或为 0。		
算法讨论	令所有数的二进制位数最多为 m，则任意选择 $m+1$ 个数一定能找到一个非空子集使得子集内的数异或为 0。可以每次随机选出 $m+1$ 个数，用高斯消元求出一组解，将这些数删除并重复。 在求出一组解时应该选择数尽可能少的解。可以多次随机自由变量的值，找到一组尽可能小的解。		
时空复杂度	时间复杂度 $O(nm^2+nmt)$，空间复杂度 $O(n)$，其中 t 为寻找最小解时尝试的次数。		

试题编号	FAULT	试题名称	Fault Tolerance
题目大意	给定一个模 2 意义下的矩阵，要求删除尽可能少的行使得列不满秩。		
算法讨论	显然删除某一列为 1 的所有行是一种可行解。注意到将一列异或上另一列不会改变答案，可以每次随机将一列异或上另一列，并更新答案。每次可以选择某一列中的一个 1，将剩下所有列中这一位为 1 的消去，更新答案。		
时空复杂度	时间复杂度 $O(tnm)$，空间复杂度 $O(nm)$，其中 t 为尝试的次数。		

试题编号	CHAORNOT	试题名称	To challenge or not
题目大意	给定一些整数，你需要找出尽可能多的数，使得任意三个数不构成等差数列。		
算法讨论	首先贪心得到一组解。维护和已经加入的数形成等差数列不超过一个的数。显然加入这些数并删除形成等差数列的数后答案不会变差。可以用模拟退火解决。用形成等差数列不超过一个的数的个数来判断答案的优劣。		
时空复杂度	时间复杂度 $O(n^2\log n+tn\log n)$，空间复杂度 $O(n)$，其中 t 为模拟退火的步数。		

试题编号	SEASNAKE	试题名称	Sereja and Snake
题目大意	给定一个贪吃蛇游戏，要求用尽可能少的步数吃完所有食物。		
算法讨论	显然一种做法是沿着一条哈密尔顿回路走。这样一定能够吃完所有食物，但是当蛇的长度较小时不优。假设选择的哈密尔顿回路为依次遍历每行，则可以考虑跳过几行直接从后面继续。为了保证仍然能找到解，可以再按照哈密尔顿回路模拟一个周期判断是否有无解情况。 注意到可以将棋盘翻转后重新计算，得到最优的一组解。		
时空复杂度	时间复杂度 $O(n^3m^3)$，空间复杂度 $O(nm)$。		

试题编号	SEAVEC	试题名称	Sereja and Vectors
题目大意	给定一些向量，要求选出一些向量，使得这些向量之和的每一维不超过另一个向量的对应维。需要使得选出的向量个数尽可能多，向量之和尽可能接近给定的向量。		
算法讨论	首先考虑一个贪心算法。将向量按照大小排序，每次选择最小的一个尝试加入。向量的大小可以用每一维的平方和来表示。 接下来用模拟退火。每次尝试加入一个向量并删除一个向量。注意到有可能更新答案会很慢，可以记录每次修改答案的方式，这样更新答案时只需要重新做这些操作。		
时空复杂度	时间复杂度 $O(n\log n + nk + tk)$，空间复杂度 $O(n + k)$，其中 t 为模拟退火的步数。		

试题编号	SEAPERM	试题名称	Sereja and Permutation
题目大意	给定一个序列。你需要求出这个序列的一个排列，使得从前 k 个位置开始分别在满足和不超过 s 的条件下向右延伸，使得所有的和尽可能大。		
算法讨论	首先考虑一种贪心。可以从小到大加入数，并且对于每个位置向右延伸的最后一个位置，可以在后面选择一个使答案最优的数替换这个数。但是这种贪心对于某些情况不优。再考虑另一种贪心。将所有数随机打乱，每次选择一个合法且使答案最优的数加入到后面。重复多次可以对某些数据得到很好的解。 然后考虑用模拟退火，每次将一个区间内的数随机打乱。这样可以通过本题。		
时空复杂度	时间复杂度 $O(t_1 nk + t_2 n)$，空间复杂度 $O(n)$，其中 t_1 为尝试贪心的次数，t_2 为模拟退火的步数。		

试题编号	GERALD09	试题名称	Chef and Rectangle Genome
题目大意	给定 n 和 m，你需要构造出一个 $n \times m$ 的只包含 GCAT 四种字符的矩阵，使得不相同的子矩阵数量尽可能接近 k。		
算法讨论	考虑在一个解的基础上随机改变一些位置，并判断是否更优。可以将改变的概率设为递减，这样可以很快接近 k。注意到每次随机很难让答案变小，可以限制答案不能太大，否则可能难以接近 k。		
时空复杂度	时间复杂度 $O(tn^2m^2)$，空间复杂度 $O(nm)$，其中 t 为尝试的次数。		