数学

Amagi_Yukisaki

线性代数

行列式，矩阵乘法，快速幂，高斯消元，Matrix-Tree

行列式

- 行列式在数学中，是一个函数，其定义域为det的矩阵A，取值为一个标量，写作$\det(A)$或$|A|$。
- 对于一个$n*n$的矩阵A，我们枚举n个数的所有排列p，定义其逆序数为$r(p)$。
- 则有：$|A| = \sum_p (-1)^{r(p)} \prod_{i=1}^{n} a_{i,p_i}$。

行列式

- 求法：
- 高斯消元把矩阵消成一个上三角，主对角线上的元素乘积即为答案。
- 记得记录你交换两行的次数，次数为奇则取负。
- 如果主对角线上有0，则说明原矩阵中存在两行线性相关，故行列式也应该为0。

矩阵乘法

- 大小为$n*m$的矩阵A可以和大小为$m*p$的矩阵B相乘得到大小为$n*p$的矩阵C。
- $C_{i,j}=\sum_{k=1}^{m}A_{i,k}*B_{k,j}$。
- 一般我们的循环顺序为i,j,k,。
- 优化为i,k,j有奇效，记录下$A_{i,k}$优化缓存。
- 《深入理解计算机系统》上说能快2倍。
- 相当于n个点走到m个中间点的邻接矩阵，和m个中间点走到p个点的邻接矩阵，得出的答案就是从左边n个点到右边p个点的路径方案数。

矩阵乘法

- 特殊的矩阵乘法：
- min转移：
- $C_{i,j} = \sum_{k=1}^{m} min(\{A_{i,k} + B_{k,j}\})$
- 就是所有方案中取min，类似最短路。
- max转移：
- $C_{i,j} = \sum_{k=1}^{m} max(\{A_{i,k} + B_{k,j}\})$
- 就是所有方案中取max，类似最长路。

快速幂

- 本质就是二进制分解。
- 把矩阵写成一个$class$，直接带入普通快速幂即可。
- 只适用于方阵。
- 注意”1”矩阵的构造。
- 对于min转移和max转移，可以构造出全为inf 的矩阵和全为$-inf$ 的矩阵。

高斯消元

- 用于求方程的解和矩阵的行列式。
- 依次为每个变量找到代表方程，将该变量在这个方程中的系数置为1，用这个方程去消除其他方程中的这个变量。
- 注意增广矩阵中的常数列也要参与运算。

Matrix-Tree

- 一个无向图生成树的数量为：
- 它的拉普拉斯矩阵（度数-邻接）任意一个代数余子式（去掉一行一列后的行列式）的值。
- 有向图？一定删除掉作为起点的那个点。
- 入度矩阵内向树$(u->v,++A_{u,v})$，出度矩阵外向树。注意有向图没有邻接矩阵。

Matrix-Tree

- 拉普拉斯矩阵的任意一个代数余子式都是相等的。
- （因为它线性相关，去掉任意一行一列后的所有矩阵都能相互推出）
- 根据柯西-比内公式，有：

$$\det(AB) = \sum_{S \in \binom{[n]}{m}} \det(A_{[m],S}) \det(B_{S,[m]}).$$

- 就是计算$n * m, m * n$的矩阵相乘得到的矩阵的行列式，可以从m列中每次枚举一个大小为n的子集，再从n行中取出相同的子集，计算两个方阵的行列式，相乘。求和即可。

Matrix-Tree

- 我们将n个点m条边的图中的m条边建立为中间点。
- 定义$n*m$的矩阵A，如果点i和边j相连，若$i<j$，则$A_(i,j)=1$；若$i>j$，则$A_(i,j)=-1$。
- 定义矩阵B为A行列交换后的结果。（按照主对角线对称）
- 则整张图的拉普拉斯矩阵C即为$A*B$。
- 考虑柯西-比内公式的组合意义。
- 我们计算C的某个代数余子式的时候，相当于在A的m列中取出一个大小为$n-1$的子集，再从B的m列中取出一个大小为$n-1$的子集。
- 将两个方阵相乘，计算行列式。（$C=A*B,|C|=|A|*|B|$）

Matrix-Tree

- 因为两个方阵互为转置，所以其相乘的行列式为某个方阵行列式的平方。
- 考虑计算单独一个方阵的行列式。
- 如果我们得到n个点$n-1$条边的图中有环，则最终得到的矩阵会线性相关，其行列式为0。
- 是树的话，其行列式为-1或$+1$。（逐行列添加元素，数学归纳法证明）
- 故每种生成树的方案均会被统计一次。

组合数学

概率和期望，计数，容斥

概率和期望

- 概率：
- 对于随机事件A，概率$P(A)$表示事件A发生的概率。
- 期望：
- 对于简单随机变量X，期望$E(X) = \sum X_i * P(X_i)$。
- 可理解为所有情况的平均值

计数，容斥

- 考虑求满足n个条件的方案数目。
- 我们能求破坏某组条件S的方案数目，定义为a_S，同时S中的条件数目为r_S
- 则所求答案为：$\sum_S (-1)^{r_S} * a_S$。
- 这个公式看起来不知道怎么用。需要从题目中理解。

题目

通过实际应用理解基础知识

BZOJ4031: [HEOI2015]小Z的房间

- 你突然有了一个大房子，房子里面有一些房间。事实上，你的房子可以看做是一个包含n*m个格子的格状矩形，每个格子是一个房间或者是一个柱子。在一开始的时候，相邻的格子之间都有墙隔着。你想要打通一些相邻房间的墙，使得所有房间能够互相到达。在此过程中，你不能把房子给打穿，或者打通柱子（以及柱子旁边的墙）。同时，你不希望在房子中有小偷的时候会很难抓，所以你希望任意两个房间之间都只有一条通路。现在，你希望统计一共有多少种可行的方案。
- 一行一个整数，表示合法的方案数 Mod 10^9
- 对于前100%的数据，n,m<=9

BZOJ4031: [HEOI2015]小Z的房间

- 本来这题的正解不是Matrix-Tree，因为取模的数不是质数。
- 然而通过奇怪的科技能用Matrix-Tree水过去。
- 正常建立图，相邻格子连边。
- 然后高斯消元。因为我们没有除法这种操作，所以需要通过类似gcd的辗转相除手段进行消元，直到某一行上当前主元的系数消为0为止。
- 根据gcd 算法，显然一定能消成为0。

BZOJ3640: JC的小苹果

- 开始JC在1号点，他的小苹果在N号点。DZY在一些点里放了怪兽。当JC每次遇到位置在i的怪兽时他会损失Ai点血。当JC的血小于等于0时他就会被自动弹出迷宫并且再也无法进入。
- 但是JC迷路了，他每次只能从当前所在点出发等概率的选择一条道路走。所有道路都是双向的，一共有m条，怪兽无法被杀死。现在JC想知道他找到他的小苹果的概率。
- 对于100%的数据 2<=n<=150，hp<=10000，m<=5000，保证图联通。

BZOJ3640: JC的小苹果

- 首先能想到暴力DP：
- $f_{i,j}$表示在剩余血量为i，当前在点j的期望。如果存在边j,k，则转移为：$f_{i-w[k],k} \mathrel{+}= \frac{f_{i,j}}{deg_i}$。
- 当$w[k]$为1时，直接转移。
- 当$w[k]$为0时，需要层内高斯消元。
- 每次转移只有常数项不同，其余操作相同。
- 高斯消元的过程可以看右乘以一个转移矩阵A。
- 列出层内的转移，进行高斯消元，同时对单位矩阵进行相同的操作，得到矩阵A即可。

BZOJ5300: [Cqoi2018]九连环

- 九连环是一种源于中国的传统智力游戏。
- 1．第一个（最右边）环任何时候都可以任意装上或卸下
- 2．如果第k个环没有被卸下，且第k个环右边的所有环都被卸下，则第k+l个环（第k个环左边相邻的环）
- 可以任意装上或卸下与魔方的千变万化不同，解九连环的最优策略是唯一的。
- 随着环数增加，需要的步数也会随之增多。例如卸下九连环．就至少需要341步。
- 请你计算，有n个环的情况下，按照规则，全部卸下至少需要多少步。

BZOJ5300: [Cqoi2018]九连环

- 手玩得到其奇数项的转移矩阵为：
- $f_n = 4f_{n-2} + 1$。
- 矩阵乘法优化转移。
- 偶数项为前一个奇数项$* 2$。
- 因为答案很大，所以需要FFT优化。

BZOJ3143: [Hnoi2013]游走

- 一个无向连通图，顶点从1编号到N，边从1编号到M。
  小Z在该图上进行随机游走，初始时小Z在1号顶点，每一步小Z以相等的概率随机选 择当前顶点的某条边，沿着这条边走到下一个顶点，获得等于这条边的编号的分数。当小Z 到达N号顶点时游走结束，总分为所有获得的分数之和。
  现在，请你对这M条边进行编号，使得小Z获得的总分的期望值最小。
- 输入保证30%的数据满足N≤10，100%的数据满足2≤N≤500且是一个无向简单连通图。

BZOJ3143: [Hnoi2013]游走

- 高斯消元建图计算出到达每个点的次数的期望。
- 对于边u,v，经过这条边次数的期望就是点u的次数期望/点u的度数+点v的次数期望/点v的度数。
- 逆序排列即可。

BZOJ4008: [HNOI2015]亚瑟王

- 玩家有一套卡牌，共 n张。游戏时，玩家将 n 张卡牌排列成某种顺序，排列后将卡牌按从前往后依次编号为 1 ~ n。本题中，顺序已经确定，即为输入的顺序。每张卡牌都有一个技能。第 i 张卡牌的技能发动概率为 pi，如果成功发动，则会对敌方造成di点伤害。也只有通过发动技能，卡牌才能对敌方造成伤害。 0 < pi < 1。
- 一局游戏一共有 r 轮。在每一轮中，系统将从第一张卡牌开始，按照顺序依次考虑每张卡牌。在一轮中，对于依次考虑的每一张卡牌：
- 1如果这张卡牌在这一局游戏中已经发动过技能，则
- 1.1 如果这张卡牌不是最后一张，则跳过之（考虑下一张卡牌）； 否则（是最后一张），结束这一轮游戏。
- 2否则（这张卡牌在这一局游戏中没有发动过技能），设这张卡牌为第 i 张
- 2.1将其以 pi的概率发动技能。
- 2.2如果技能发动，则对敌方造成 di点伤害，并结束这一轮。
- 2.3如果这张卡牌已经是最后一张（即 i 等于n），则结束这一轮；否则，考虑下一张卡牌。
- 请帮助小 K 求出这一套卡牌在一局游戏中能造成的伤害的期望值。
- 1 <= T <= 444， 1 <= n <= 220， 0 <= r <= 132， 0 < pi < 1， 0 <= di <= 1000。

BZOJ4008: [HNOI2015]亚瑟王

- $f_{i,j}$表示考虑前i张牌，存在j轮能轮到第i张牌判定的期望。
- 则转移为：
- $f_{i,j} = f_{i-1,j} * (1 - p_{i-1})^j + f_{i-1,j+1} * (1 - (1 - p_{i-1})^{j+1})$
- 前者表示第$i-1$张牌在剩余j次机会时没有打出去，后者表示第$i-1$张牌在剩余$j+1$次机会时打出去了。
- 答案即为：$\sum_{i=1}^{n} \sum_{j=1}^{r} f_{i,j} * \left(1 - (1 - p_i)^j\right) * d_i$

BZOJ4872: [Shoi2017]分手是祝愿

- Zeit und Raum trennen dich und mich.
- 时空将你我分开。B 君在玩一个游戏，这个游戏由 n 个灯和 n 个开关组成，给定这 n 个灯的初始状态，下标为
- 从 1 到 n 的正整数。每个灯有两个状态亮和灭，我们用 1 来表示这个灯是亮的，用 0 表示这个灯是灭的，游戏的目标是使所有灯都灭掉。但是当操作第 i 个开关时，所有编号为 i 的约数（包括 1 和 i）的灯的状态都会被改变，即从亮变成灭，或者是从灭变成亮。B 君发现这个游戏很难，于是想到了这样的一个策略，每次等概率随机操作一个开关，直到所有灯都灭掉。这个策略需要的操作次数很多， B 君想到这样的一个优化。如果当前局面，可以通过操作小于等于 k 个开关使所有灯都灭掉，那么他将不再随机，直接选择操作次数最小的操作方法（这个策略显然小于等于 k 步）操作这些开关。B 君想知道按照这个策略（也就是先随机操作，最后小于等于 k 步，使用操作次数最小的操作方法）的操作次数的期望。这个期望可能很大，但是 B 君发现这个期望乘以 n 的阶乘一定是整数，所以他只需要知道这个整数对 100003 取模之后的结果。

BZOJ4872: [Shoi2017]分手是祝愿

- 首先每次开关都是不同的，即：不能通过操作某个不含开关A的开关的集合来实现开关A的效果。
- 之后就很简单了。由于一个操作开关X只能影响$\leq X$的灯，所以我们按照编号从大到小，逐个操作每个亮着的灯。这样的操作次数即为初始局面下需要的正确操作次数。
- 令f_i表示当前状态需要i次正确操作，所需要操作次数的期望。
- 显然$f_k = 0$，且当需要i次正确操作时，有i个开关是被正确操作所包含的。
- 所以：$f_i = \frac{i}{n} f_{i-1} + \frac{n-i}{n} f_{i+1} + 1$

BZOJ4872: [Shoi2017]分手是祝愿

- 整理一下，我们有：
- $nf_i = if_{i-1} + (n-i)f_{i+1} + n$
- $i(f_{i+1} - f_{i-1}) = n(f_{i+1} - f_i) + n$
- 定义$g_i = f_{i+1} - f_i$
- 则有：
- $i(g_i + g_{i-1}) = ng_{i-1} + n$
- 现在g可以递推了，问题解决。
- 最后注意要乘阶乘！

BZOJ3566: [SHOI2014]概率充电器

- SHOI 概率充电器由 n-1 条导线连通了 n 个充电元件。进行充电时,每条导线是否可以导电以概率决定,每一个充电元件自身是否直接进行充电也由概率决定。
  随后电能可以从直接充电的元件经过通电的导线使得其他充电元件进行间接充电。
  作为 SHOI 公司的忠实客户,你无法抑制自己购买 SHOI 产品的冲动。在排了一个星期的长队之后终于入手了最新型号的 SHOI 概率充电器。
  你迫不及待地将 SHOI 概率充电器插入电源——这时你突然想知道,进入充电状态的元件个数的期望是多少呢?
- 对于 100%的数据,n≤500000,0≤p,qi≤100。

BZOJ3566: [SHOI2014]概率充电器

- 正难则反，f_i表示考虑以节点i为根的子树，不能给i供电的概率。
- 则$f_i = (1 - q_i) * \prod_{son_i}(f_{son_i} + (1 - f_{son_i})(1 - p_{e_{son_i}}))$
- g_i表示考虑i的父亲节点以及其他儿子，不能给i供电的概率。
- $g_i = 1 - p_{e_i}(1 - g_{fa_i} * \frac{f_{fa_i}}{f_i + (1 - f_i)(1 - p_{e_i})})$
- 答案即为：$\sum_{i=1}^{n} 1 - f_i * g_i$

BZOJ4767: 两双手

- 老W是个棋艺高超的棋手，他最喜欢的棋子是马，更具体地，他更加喜欢马所行走的方式。老W下棋时觉得无聊，便决定加强马所行走的方式，更具体地，他有两双手，其中一双手能让马从(u,v)移动到(u+Ax,v+Ay)而另一双手能让马从(u,v)移动到(u+Bx,v+By)。小W看见老W的下棋方式，觉得非常有趣，他开始思考一个问题：假设棋盘是个无限大的二维平面，一开始马在原点(0,0)上，若用老W的两种方式进行移动，他有多少种不同的移动方法到达点(Ex,Ey)呢？两种移动方法不同当且仅当移动步数不同或某一步所到达的点不同。老W听了这个问题，觉得还不够有趣，他在平面上又设立了n个禁止点，表示马不能走到这些点上，现在他们想知道，这种情况下马有多少种不同的移动方法呢？答案数可能很大，你只要告诉他们答案模(10^9+7)的值就行。
- |Ax|,|Ay|,|Bx|,|By| <= 500, 0 <= n,Ex,Ey <= 500

BZOJ4767: 两双手

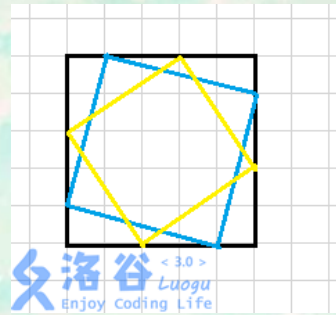
- 首先从原点走到点(x,y)的所有路径中，两种走法的数量是一定的。可以通过二元一次方程组求解。
- 如果两种走法各自需要a,b次，那么方案数为：$\binom{a+b}{a}$。
- 将所有坏点标记为关键点，同时将终点也标记为关键点。
- f_i表示从原点到点i，第一个关键点为点i的方案数。
- 则$f_i = walk(0,i) - \sum_{j=1,j\neq i}^{n} walk(j,i) * f_j$
- 注意每个点只会从其左上方的点转移。
- 答案即为f_{n+1}，其中第$n+1$个点为终点。

BZOJ4558: [JLoi2016]方

- 上帝说，不要圆，要方，于是便有了这道题。由于我们应该方，而且最好能够尽量方，所以上帝派我们来找正方形上帝把我们派到了一个有N行M列的方格图上，图上一共有(N+1)×(M+1)个格点，我们需要做的就是找出这些格点形成了多少个正方形（换句话说，正方形的四个顶点都是格点）。但是这个问题对于我们来说太难了，因为点数太多了，所以上帝删掉了这(N+1)×(M+1)中的K个点。既然点变少了，问题也就变简单了，那么这个时候这些格点组成了多少个正方形呢?
- 0 <=x <=N<=10^6, 0 <=y<=M<=10^6，K<=2*1000且不会出现重复的格点。

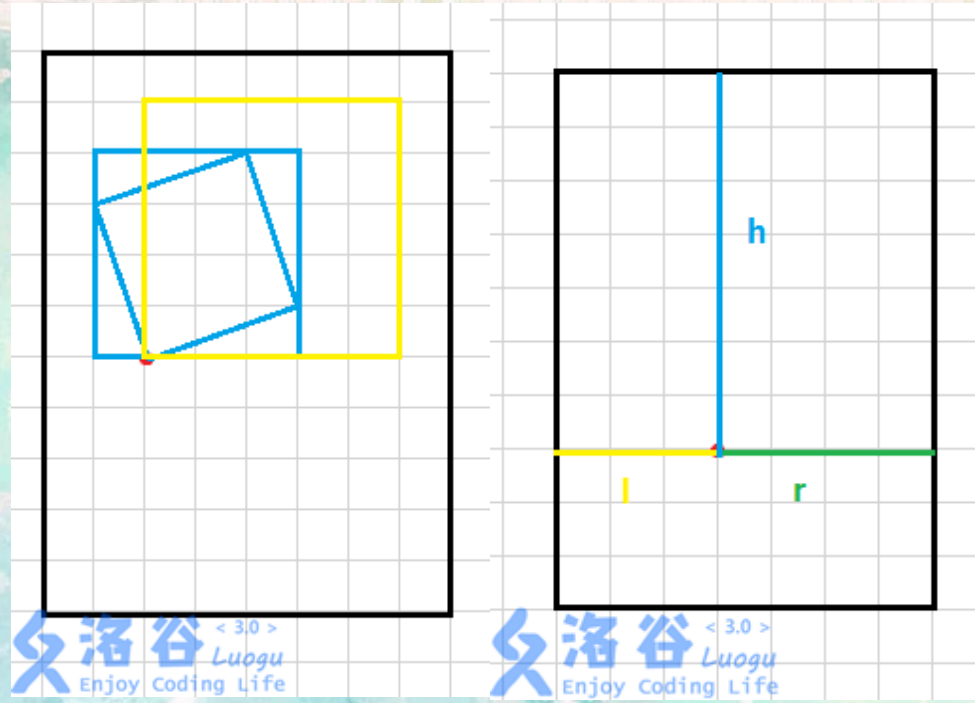
BZOJ4558: [JLoi2016]方

- 注意正方形可以是斜着放的……
- 容斥，显然答案为至少含偶数个坏点的-至少含奇数个坏点的。
- 考虑至少含0个坏点的。
- 对斜着的正方形找到一个最小的与轴平行的正方形将其放入。
- 之后枚举外部正方形的边长。答案为：
- $\sum_{i=1}^{\min(n,m)}(n-i+1)*(m-i+1)*i$
- （无视洛谷水印，我抄我自己？



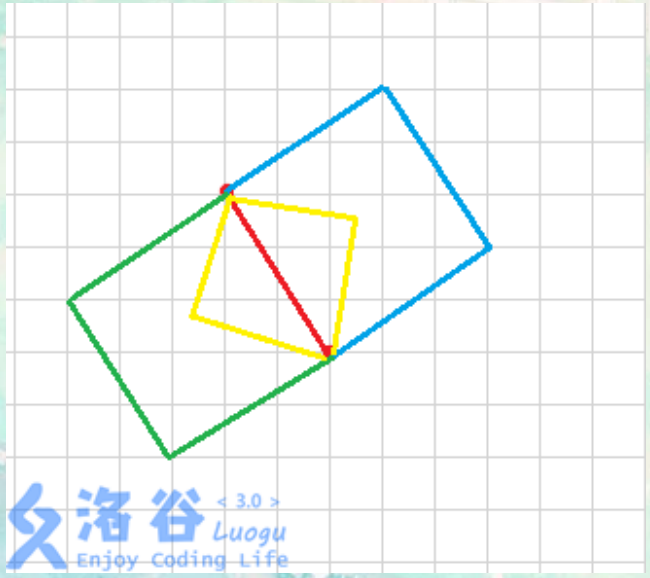
BZOJ4558: [JLoi2016]方

- 考虑至少含一个坏点的情况。
- 此时这个坏点一定在该正方形外部的最小的与轴平行的正方形的角上或边上。
- 令$t = \min(l + r, h)$，手玩一下发现
- 此时能取到的位置数量为
- 当$t \le l$且$t \le r$时，$t * (t + 3)/2$
- 当$t > l$时，这样算会越过左边界。
- 应该减去的为：$(t - l) * (t - l + 1)/2$
- 右边界同理。



BZOJ4558: [JLoi2016]方

- 考虑包含两个以上坏点的，如果有两个坏点，则我们可以确定一个正方形。只有三种情况（黄色情况的合法性需要判定）：
- 之后我们分别计算三种情况和每种情况的坏点数，可以同时统计出含三个、四个坏点的情况。



BZOJ5332: [Sdoi2018]旧试题

- 计算如下表达式的值。
- $\sum_{i=1}^{A}\sum_{j=1}^{B}\sum_{k=1}^{C}d(ijk)$
- 其中$d(ijk)$表示$i*j*k$的约数个数。
- 对于 100 分的数据，$1 \le T \le 10, 1 \le A, B, C \le 10^5, 1 \le \sum \max(A, B, C) \le 2 * 10^5$。

BZOJ5332: [Sdoi2018]旧试题

$$\sum_{i=1}^{A}\sum_{j=1}^{B}\sum_{k=1}^{C} d(ijk)$$

$$=\sum_{i=1}^{A}\sum_{j=1}^{B}\sum_{k=1}^{C}\sum_{x|i}\sum_{y|j}\sum_{z|k}[gcd(x,y)=1][gcd(y,z)=1][gcd(z,x)=1]$$

$$=\sum_{x=1}^{\max(a,b,c)}\sum_{y=1}^{\max(a,b,c)}\sum_{z=1}^{\max(a,b,c)}[gcd(x,y)=1][gcd(y,z)=1][gcd(z,x)=1]\frac{A}{x}\frac{B}{y}\frac{C}{z}$$

$$=\sum_{x=1}^{\max(a,b,c)}\sum_{y=1}^{\max(a,b,c)}\sum_{z=1}^{\max(a,b,c)}\frac{A}{x}\frac{B}{y}\frac{C}{z}\sum_{u|x,u|y}\sum_{v|y,v|z}\sum_{w|z,w|x}\mu(u)\,\mu(v)\mu(w)$$

$$=\sum_{x=1}^{\max(a,b,c)}\sum_{y=1}^{\max(a,b,c)}\sum_{z=1}^{\max(a,b,c)}\frac{A}{x}\frac{B}{y}\frac{C}{z}\sum_{u|x,u|y}\sum_{v|y,v|z}\sum_{w|z,w|x}\mu(u)\,\mu(v)\mu(w)$$

$$=\sum_{u=1}^{\max(a,b,c)}\sum_{v=1}^{\max(a,b,c)}\sum_{w=1}^{\max(a,b,c)}\mu(u)\mu(v)\mu(w)\left(\sum_{lcm(u,v)|x}\frac{A}{x}\right)*\left(\sum_{lcm(v,w)|y}\frac{B}{y}\right)*\left(\sum_{lcm(w,u)|z}\frac{C}{z}\right)$$

BZOJ5332: [Sdoi2018]旧试题

- 后面的三个求和号内的内容可以暴力预处理。
- 前面三个求和号的内容，让当数a,b的LCM小于限制时候，度数小的向度数大的连边。
- 然后就是数三元环啦。

BZOJ2829 集合计数

- 一个有N个元素的集合有2^N个不同子集（包含空集），现在要在这2^N个集合中取出若干集合（至少一个），使得
- 它们的交集的元素个数为K，求取法的方案数，答案模1000000007。（是质数喔~）

BZOJ2829 集合计数

- 我们会求交集至少含k个元素，$k+1$个元素…n个元素的集合的数量。
- 依旧多偶数个则加，多奇数个则减。
- 考虑至少有i个元素的集合对答案的贡献。
- 从n个元素中选出i个，表示当前至少相同的。
- 从i个元素中选出k个，表示当前的集合在至少k个元素相同的集合中被算的次数。
- 剩下的$n-i$个元素随便选，方案为：$2^{2^{n-i}}-1$。（不能没有集合）
- 剩下n-i+1个元素时，为$2^{2^{n-i+1}}-1=\left(2^{2^{n-i}}-1\right)*\left(2^{2^{n-i}}-1+2\right)$

BZOJ4036: [HAOI2015]按位或

- 刚开始你有一个数字0，每一秒钟你会随机选择一个[0,2^n-1]的数字，与你手上的数字进行或（c++,c的|,pascal的or）操作。选择数字i的概率是p[i]。保证0<=p[i]<=1，Σp[i]=1问期望多少秒后，你手上的数字变成2^n-1。
- 对于100%的数据，n<=20.

BZOJ4036: [HAOI2015]按位或

- 如果概率不为0的数or起来不能得到2^n-1，那么答案为INF。
- 考虑$\min-\max$容斥，我们有：
- $\max(S)=\sum_{T\in S}\min(T)*(-1)^{|T|-1}$
- 证明：
- 设最大值为x，对于所有的T，集合T和集合$T+x$的最小值相同，元素数量差1，故正好抵消。最后留下的只有x。

BZOJ4036: [HAOI2015]按位或

- 定义f_T为集合T中最不容易出现的元素在每一秒出现的概率的期望。
- 显然集合T中最晚出现的元素最早出现时间的期望为：$\frac{1}{f_T}$
- 而$f_T = \sum_{S \cap T \neq \emptyset} G_S$
- 只要有交就至少能在出现G时获得一个元素。
- 之后我们要求最大用时的期望，而$\frac{1}{f_T}$为最小用时的期望。$\min - \max$ 容斥即可。

RYOI2018EER：女孩子

- 好的，现在我们假设小 X 已经变成女孩子了。
- 现在，我们需要让她学习一些作为女孩子的日常呢。
- 小 X 决定把这些东西都放进一个长度为 n 的由小写英文字母组成的字符串内，她只关心这个字符串 的不重叠划分方案中，划分出模板串最多的方案所划分出的模板串个数。（例如当前字符串为abaa，模 板串为aba, a，则划分出模板串最多的方案为a/b/a/a，划分出的模板串个数个数为 3（3 个a））由于该字符串的特殊性，很多字符是不确定的，所以我们需要帮小 X 求出这个字符串的所有可能的 答案的总和取模1’000’000’009后的值。
- 当然，小 X 会时不时地确定这个字符串某一个子串必须是某些字符，你需要针对这些操作输出答 案。

RYOI2018EER：女孩子

数据范围：

测试点编号	n≤	模板串总长≤	q≤	13	1’000	25	0
1	3	10	0	14	1’000	10	50
2	3	10	0	15	1’000	10	50
3	5	10	0	16	10’000	25	0
4	5	10	0	17	10’000	25	0
5	5	10	0	18	10’000	25	0
6	10	10	0	19	10’000	10	50
7	10	10	0	20	10’000	10	50
8	10	10	30	21	1’000’000’000	25	100
9	10	10	30	22	1’000’000’000	25	100
10	10	10	30	23	1’000’000’000	25	100
11	1’000	25	0	24	1’000’000’000	25	100
12	1’000	25	0	25	1’000’000’000	25	100

保证每次确定的子串长度总和 $\leq 1'000$且每次确定的子串长度 ≤ 50（虽然第二个条件没用）。

前 20 个测试点的时间限制为 1s，最后 5 个测试点的时间限制为 5s。

RYOI2018EER：女孩子

- 对于$n \leq 5$的测试点，我会爆搜！
- 把模板串插入一个AC自动机内，大力搜索+匹配即可。
- 对于$n \leq 10'000$的测试点，我会DP！
- 我们记录状态$pair < int, int >$ $f[i][j]$表示考虑完前i个字符，在自动机节点j，方案数和总匹配次数。
- 转移的话，枚举一下下一个字符是那个，暴力转移即可。对于被钦定的字符，只枚举那一种即可。
- 复杂度$O(26nq\sum len)$
- 好的，现在你获得了80分。
- (毕竟我还有一个艰巨的任务叫给大家送分，~~不过送80分有点过分了吧~~)

RYOI2018EER：女孩子

- **前方高能预警！**
- 对于$n \leq 1'000'000'000$的数据呢？
- 看到这种数据范围，不是倍增就是矩乘。这道题的话，就是矩乘了。这个DP的转移易写成矩阵形式。
- 但是只是矩乘还不行，我们需要一个叫做“正反自动机状态合并”的科技。
- 考虑我们把模板串正着反着分别插入两个AC自动机内，会发生什么呢？
- 反向的那个自动机，节点记录的是：**最长的有可能成为某个字符串的后缀**的子串。
- 我们用两个自动机从前从后分别DP，在中间进行合并。
- 两个节点状态已经完成的完整匹配，直接统计答案即可。关键是跨越分割点的匹配。
- 暴力读出反向自动机表示的字符串，在第一个自动机上走这个字符串，看能否走到结束节点，如果能的话，统计答案。（显然这样走最多经过一次结束节点，否则反向自动机会直接把后一半记录为一个完整匹配）

RYOI2018EER：女孩子

- 其实吧……上一页幻灯片给的那个做法，完全是我石乐志。（虽然我的std的确是这么写的）
- 实际上我们只需要一个自动机，考虑我们优化一开始的那个DP思路：首先是一大段完全相同的转移，然后是一小段每个字符不同的转移，然后又是一大段完全相同的转移。这样的话，我们只需要把DP的转移方程写成矩阵形式，快速幂优化那两段完全相同的转移就好了。
- 为什么会想到自动机合并呢？因为我石乐志……
- 大家可以做一下$NTOJ1010$: $[RYOI2018]$如月红，实际上这题是它的无敌弱化版……原版似乎是必须用自动机的合并科技的。

RYOI2018EER：女孩子

- 对了，这题还有一点细节：
- 观察这样一个示例：模板串为acc和c。显然acc无论如何是没有用的，因为出现一个acc的地方会出现两个c，而两个c更优。所以，我们在走到acc的c上时，需要把这个c也标记为结束节点。
- 另外，我们需要把结束节点的所有出边指向根节点对用出边的节点，因为这里要求不重复匹配。
- 所以这个$getFail()$应该这样写：

```
inline void buildFail() {
    queue<int> q;
    for(int i=0;i<26;i++) ( ch[root][i] ? q.push(ch[root][i]) , fail[ch[root][i]] : ch[root][i] ) = root;
    while( q.size() ) {
        const int pos = q.front(); q.pop() , isend[pos] |= isend[fail[pos]];
        for(int i=0;i<26;i++) ( ch[pos][i] ? q.push(ch[pos][i]) , fail[ch[pos][i]] : ch[pos][i] ) = ch[fail[pos]][i];
    }
    for(int i=1;i<=cnt;i++) if( isend[i] ) for(int j=0;j<26;j++) ch[i][j] = ch[root][j];
}
```

~~(话说我的样例还是挺良心的是吧，原本第二个样例就是acc和c，怕大家调不出来心态爆炸，于是就改成了ac和c)~~

RYOI2018R2：推 gal 自动机

- 小 X 是一个沉迷 galgame 的死宅。 有一天他在攻略一款游戏时，发现这款游戏有一个可怕的无限循环。 如果要逃出这个循环，便需要回答 n 个问题。
- 这些问题的选项数量十分特殊。每个问题的选项数量与问题本身无关，而是和这个问题 是第几个被回答的有关。 形式化的说法，如果当前还有 i 个问题需要被回答，那么游戏将为这个问题提供 ci 个选项。
- 小 X 发现这个游戏还有一些神奇的特性。如果当前还有 i 个问题需要回答，如果回答正 确，则继续回答下一个问题； 而如果回答错误，则游戏会从区间[i,n]随机生成一个需要回答问题的数量 size，然后从 n 个问题中随机选择 size 个问题并打乱答案，让小 X 继续回答。
- 由于小 X 快进了不少剧情，他当然是不知道这些问题的正确答案的。 于是他写了一个程序来回答问题，也就是每次无脑选择第一个选项！
- 他想知道从开始(剩余n个问题)到剩余q个问题的状态，这个程序期望进行多少次选择。 当然这个数字不一定是个整数，所以请告诉小 X 这个数字在 mod 1’000’000’009 意义下 的值。 由于小 X 的程序存在未知 bug，有时候会造成游戏的 ci 值修改。而你，需要回答修改后 询问的值。
- n<=100000,m<=150000

RYOI2018R2：推 gal 自动机

- 显然期望DP最重要的就是状态转移方程。没有状态转移方程的话，什么都做不了啊。
- 我们定义f_i为从还剩下i个问题没有回答的状态，到我们的期望状态，所需要的步数。
- 那么我们有：$f_i = \frac{f_{i-1}}{c_i} + \frac{c_i - 1}{c_i} \frac{1}{n-i+1} \sum_{j=i}^{n} f_j + 1$
- 显然对于每次询问，我们期望从开始达到剩余q个问题的状态的话，我们有$f_q = 0$，而我们的答案，就是这个条件下的f_n。

RYOI2018R2：推 gal 自动机

- 有了这个东西我们能干什么呢？我们能每次进行高斯消元，这样能通过6个测试点。
- 不过这样也太没有梦想了点……
- 我们不妨设$f_n = x$，显然我们可以用$kx + b$表示所有的f_i。
- 这样我们能一次高斯消元求出所有的所有的f_i，询问的时候直接带入$f_q = 0$求解x，只用在修改的时候全部重构。这样又能通过3个测试点了。

RYOI2018R2：推 gal 自动机

- 等等，这么优美的公式为什么要高斯消元呢?
- 我们对转移方程进行变形：
- $f_i = \frac{f_{i-1}}{c_i} + \frac{c_i - 1}{c_i} \frac{1}{n-i+1} \sum_{j=i}^{n} f_j + 1$
- $c_i f_i = f_{i-1} + (c_i - 1) \frac{1}{n-i+1} \sum_{j=i}^{n} f_j + c_i$
- 假设我们已知了$f_{[i,n]}$，我们能否用这个公式推出f_{i-1}呢?
- $f_{i-1} = c_i f_i - (c_i - 1) \frac{1}{n-i+1} \sum_{j=i}^{n} f_j - c_i$
- 好的，我们又能通过4个测试点了。

RYOI2018R2：推 gal 自动机

- 我们考虑后面的那个Σ是不是能优化成一个后缀和啊。
- $f_{i-1}=c_i f_i-(c_i-1)\frac{1}{n-i+1}\sum_{j=i}^{n} f_j-c_i$
- 我们替换下标
- $f_i=c_{i+1} f_{i+1}-(c_{i+1}-1)\frac{1}{n-i}\sum_{j=i+1}^{n} f_j-c_i$
- 我们定义：$s_i=\sum_{j=i}^{n} fj$
- 则我们有：
- $f_i=c_{i+1} f_{i+1}-(c_{i+1}-1)\frac{1}{n-i} s_{i+1}-c_{i+1}$
- 这样就可以$O(n)$转移了，又能通过6个测试点。

RYOI2018R2：推 gal 自动机

- 观察这个转移，我们发现它是线性的，且i的状态只和$i+1$的状态有关。
- 我们能否把他写成矩阵呢?

```
fi+1.k              fi+1.b              si+1.k              si+1.b              1
*
ci+1                0                   ci+1                0                   0
0                   ci+1                0                   ci+1                0
-(ci+1-1)/n-i       0                   -(ci+1-1)/n-i+1     0                   0
0                   -(ci+1-1)/n-i       0                   -(ci+1-1)/n-i+1     0
0                   -ci+1               0                   -ci+1               1
=
fi.k                fi.b                si.k                si.b                1
```

你们就自行脑补我写的这是矩阵就好了，注意所有的c都是c_{i+1}，此外所有的$n-i$都为最先计算。

RYOI2018：优化

[RYOI2018]优化(Optimize)

Description

小X又接到了一个十分有趣的活:优化galgame的效率。

公司为了统计各位玩家对游戏的喜爱程度，将玩家对游戏的喜爱度进行了量化。

而k阶喜爱度被定义为sigma(x=1,n)(sigma(i1=1,x)sigma(i2=1,x)...sigma(ik=1,x))gcd(x,i1,i2,...ik)。

$$\sum_{x=1}^{n}\underbrace{\sum_{i1=1}^{x}\sum_{i2=1}^{x}\dots\sum_{ik=1}^{x}}_{\text{K 层}\sum}\gcd\underbrace{(x,i1,i2,\dots,ik)}_{\text{K+1 个数的 gcd}}$$

也就是说先枚举一个上界为n的x，然后嵌套k层上界为x的求和，最终被计入答案的是这些被枚举的变量的gcd。

由于小X的同事们过于愚蠢，他们只会手写k层for循环进行计算，这样无论是编码复杂度还是算法的时间复杂度都是吃不消的。

于是小X勇敢地承接了这个任务，但是他过于蒟蒻，并不能完成。你能帮帮他吗?

Input

一行两个整数n,k。

Output

一行一个整数ans，由于答案一定很大，所以请输出答案取模998244353的值。

RYOI2018：优化

- 大力反演出phi，后面的sigma(i^k)显然是k+1次多项式，拉格朗日插值即可。


国际惯例的
谢谢大家