IOI2015国家集训队第一次作业
试题泛做

解题报告

中山纪念中学
郭隆
@*pwecar*

目录

1 Codeforces 235C Cyclical Quest

题目简述

给定一个串$S(|S| \leq 10^6)$和N个字符串$T_i(\sum |T_i| \leq 10^6)$，求对于每个串$T_i$，有多少$s$的子串是周期性同构于串$T_i$。

算法分析

我们需要求的实际上就是，对于T_i的每个周期同构的串，求出它在S中出现的次数。

对S建后缀自动机，那么走出T_i在自动机上的位置，就可以求出T_i在S中出现的次数了。要求T_i的周期同构的串在自动机上的位置，就是走多一个字符，当其parent的长度超过$|T_i|$时往回走，不断后退就可以走到其对应节点了。这一步的复杂度均摊也是线性的。

时间复杂度$O(|S| + \sum |T_i|)$。

2 Codeforces 235D Graph Game

题目简述

给定一个$N(\leq 3000)$个点N条边的无向联通图（环+外向树），对该图进行分治，每次随机选一个节点并删除该点和与该点相连的所有边，并给答案加上该点原联通块的大小。求该答案的期望。

算法分析

设$p_{a,b}$表示当a被删除时b是否还和a在同一个联通块内，那么答案就是p的和。

考虑树的情况，设a和b之间的路径上有N个点，那么$p_{a,b} = 1/N$。我们可以通过讨论删的点是否在a到b的路径上即可说明。

现在考虑经过环上的情况。设在a到环上的路径和b到环上的路径上总共有X个点，环上两条路径分别有Y和Z个点。与树的情况类似，我们分别考虑经过的路径，这样概率就是$1/(X+Y+1)+1/(X+Z+1)$。但这样会算重，因为我们此时不但要考虑删掉的点是否在当前路径上，也要考虑是否在另外一条路径上。这样就要减去$1/(X+Y+Z+1)$。

对每对点计算贡献即可。时间复杂度$O(N^2)$。

3 Codeforces 235E Number Challenge

题目简述

定义$d(n)$为n的约数个数。现在，你有三个数$a, b, c(\leq 2000)$。你的任务是计算$\sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{c} d(i*j*k) \bmod 2^{30}$的值。

算法分析

用$f_{a,b,c,p}$表示对于a, b, c来说只用了不大于p的质数的答案，则有$f_{a,b,c,p} = \sum_{i,j,k} f_{\frac{a}{p^i},\frac{b}{p^j},\frac{c}{p^k},p-1} * (i+j+k+1)$。

这样的状态数实际上不会很多，我们用hash/unordered_map来存储和转移就能通过。

4 Codeforces 238D Tape Programming

题目简述

有一个长度为$n(\leq 10^5)$的序列，每个元素可能是$0 \sim 9$之间的整数，"<"或者是">"。一开始有一个指针在序列最左端指向右方。在每个时刻如果当前位置是数字的话就输出该数字并把该数字减1，如果原本就是0的话则删除该元素。如果是"<"或者">"的话则把该指针的方向改到该符号指向的方向，如果前一个经过的位置上也是"<"或">"的话则删除原来位置的元素。在该时刻结束时指针朝着其指向的方向前进一单位，重复如上操作直到指针离开该序列为止。现在有$q(\leq 10^5)$个询问，询问对于该序列的一段子序列来说，操作结束时每个数各被输出了多少次。

算法分析

考虑从原序列的开头开始模拟操作，并且沿路上记录第i次操作经过的位置，某个位置在哪些时刻被经过了，和前i次中每个数被输出了多少次。我们可以发现，对于某个子序列$l \sim r$来说，我们可以把它看成是从原序列头开始的，只不过我们在第一次经过l时才记录答案。注意在该时刻之前，由于指针没有经过$l\ r$，这段区间里面的状态和初始状态是相同的。

确定了起点的时刻后我们还要知道这段子序列的结束时刻在原序列中的对应位置：它可能从$l-1$离开也可能从$r+1$离开。在两者中取较小值就是离开$l \sim r$的时刻。记录了前缀和之后我们就可以直接算出这段时间里面各个数输出了多少次了。

总的时间复杂度是原序列的模拟次数$+O(Q \log N)$。不难算出原序列的操作数是线性的，不会超过$20 * N$的样子。

5 Codeforces 238E Meeting Her

题目简述

城市中有$N(\leq 100)$个路口，有M条单向道路连接各个路口。有$K(\leq 100)$条公交线路，第i条线路会选择任意一条从s_i到t_i的最短路运行。问从S坐公交坐到T在最坏情况下需要转多少次车。

算法分析

设$f_{x,y}$表示现在在第y路车上经过x时最少经过多少次转车能够到达T，我们来考虑这些状态之间的转移。

设$l_1, l_2, ..., l_k$表示从x出发到达t_y只走最短路的话可能经过的下个城市，那么考虑最坏情况的话对于这些状态取个max就是了，亦就是$max(f_{l_1,y}, ..., f_{l_k,y})$。

除了继续坐下去之外，我们还可以考虑转车：在路口x下车并且换乘。但在这种情况下我们换乘的线路要一定经过x才行，否则在最坏情况下是可能不能转移到期望转移到的状态的。设$r_1, r_2, ..., r_z$表示一定经过x的线路，通过换乘能够达到的状态就是$min(f_{x,r_1}, ..., f_{x,r_z}) + 1$了。我们在两种转移中取较小值就可以求出$f_{x,y}$了。

具体实现时我们可以用类似宽搜的方法来实现。对于第一种转移我们把新状态插入队头，而对于第二种转移则把新状态插入队尾，这样就可以保证队列中的状态的f值是不递减的了。由于N和K是同阶的，总的时间复杂度是$O(N^3)$。

6 Codeforces 240E Road Repairs

题目简述

有$N(\leq 10^5)$个点$M(\leq 10^5)$条边的有向图，每条边的权值为0或1，求从以点1为根的最小树形图并输出方案。

算法分析

本题的模型是比较明显的，容易看出这题的模型就是最小树形图，也就是有向图最小生成树。

这个问题现在最常用的算法是朱刘算法，它可以求出根确定时的最小树形图。它的基本思想是对于每个点找出权值最小的一条入边，如果不构成环那么答案就被确定了。否则我们会找出一个环，把环缩起来，修改边权，然后继续重复前面的操作即可。

直接实现这个算法的复杂度是$O(NM)$的，通过一些较为复杂的技巧可以优化至$O(M \log N)$。

7 Codeforces 241B Friends

题目简述

在$N(\leq 5*10^4)$个自然数$(\leq 10^9)$中选择M对数，问每对数的异或和之和最大是多少。

算法分析

涉及到位运算的东西往往是逐位处理的，在这里我们可以贪心地从高位到低位建出01Trie后从上到下求解。

在求解每位时维护某个数当前求解到的位置（另外一个数所在的子树）。因为要异或和最大，我们显然是尽量满足取到当前位为1的对，它的数量是可以被统计出来的。如果这个数量小于M的话，我们可以算出所有该位为1的数对的贡献，具体来说就是统计子树内某一位有多少个数为1就可以了，然后M就可以减去这个对数了。如果不是的话则代表剩下的每一对都可以在该位取到1，按需移动每个数相对应的位置即可。

时间复杂度$O(N*\log^2 10^9)$。

8 Codeforces 241D Numbers

题目简述

有一个1到$N(\leq 50000)$的排列，你希望找到一个该排列的非空的子序列，使得：

1. 该子序列中的元素的异或和为0
2. 把该子序列的元素从左到右排成一列构成一个十进制的数，这个数能被$p(\leq 50000)$整除，其中p是质数

算法分析

当N比较小时我们可以直接DP，设$f_{i,j,k}$表示考虑前i个数，异或和为j，模p为k是否存在，转移和出解也很简单。时间复杂度是$O(N^2 * K)$的。

但是N可能有5万，那么该怎么办呢？一个贪心的想法是，当N太大时我们就不考虑大于某个数时的数了，比方说32或者64之类的。

这样贪心地去做的正确性是怎样的呢？不妨设N为2的整数次幂，那么在排列里取一个异或和为0的子集的概率就是$1/N$，一共有$2^N/N$个符合条件的子集。因为p是质数，我们可以认为能整除它的数的各个位的分布是随机的，对于某个数来说能整除p的概率就是$1/p$，这$2^N/N$个子集里都没有方案的概率就是$\frac{p-1}{p}^{\frac{2^N}{N}}$，当$N$取32时这个数已经远小于$10^{-5000}$了，所以我们可以认为这些集合已经可以代表所有的情况了。

9 Codeforces 241E Flights

题目简述

有一个$N(\leq 1000)$个点和$M(\leq 5000)$条有向边的DAG，你可以将每条边的权值设为1或2，要求所有从1号点到N号点的路径和相等，求方案或判断无解。

算法分析

既然我们要求每条路径的长度都相等，我们不妨设该点到N号点的距离为变量，解出方程后直接算出每条边的权值（1或2）。

那么每条边就是一个不等式：假设有一条从u到v的边，那么要满足题目中的要求，就要有$1 \leq d_u - d_v \leq 2$。这就是两条约束。而所有的约束就构成了一个差分约束系统，用Bellman-Ford或者SPFA都可以在实现内求出d。

时间复杂度$O(N * M)$。

10 Codeforces 241F Race

题目简述

有一个$N * M(N, M \leq 100)$的网格图，其中有一些格子是障碍物，有一些格子是道路，有一些格子是路口，现在给定从起点到终点沿途上经过的所有路口，问在$K(\leq 10^5)$单位时间后的位置。

算法分析

由于K比较小，我们可以考虑直接模拟走的过程。

题目中有一个比较优美的性质：给定从起点到终点沿途上经过的**所有**路口，而由于这个是网格图，我们可以通过简单的座标判断直接求出下一步前进的方向。

时间复杂度$O(K)$。

11 Codeforces 243D Cubes

题目简述

在$N * N(N \leq 1000)$的网格中，每个格子里有若干高度为1的方块，求在无穷远的地方以向量$(V_x, V_y, 0)$向网格望去能见到多少个方块。

算法分析

如果是坐标和望的方向平行的话问题的解法就很明显了：判断在某个方块前面的那些方块中最高的是多少，然后就可以知道新增了多少个看得见的方块。

不平行的话做法也是类似的：我们可以把网格旋转过来，计算在当前情况下的序，按照该序来操作。注意在旋转后挡住当前块的变成了一个区间，它所占据的位置也是一个区间，我们就需要用线段树来处理插入和查询了。

时间复杂度$O(N^2 \log N)$。

12 Codeforces 243E Matrix

题目简述

有一个$N * N(1 \leq N \leq 500)$的01矩阵a，如果它的每行的所有"1"都是连续的，那么它就被认为是好的。你可以交换任意列的顺序，构造出一个好的矩阵b并给出方案，或者判断无解。

算法分析

考虑构造一个合法解。

我们把所有的列分成若干组，每组之间的顺序是固定的，而组内之间的列之间的顺序是不定的。

每次我们考虑加入一行作为新的约束。如果加入的行有多于一组存在该组的某列在该行的取值为1的话，在加入该行后的分组情况是确定且唯一的，如果没有合法情况的话就返回无解。当某一组里并不是所有列在该行的取值都为1时，我们就需要把取值为0的列放在一起，把取值为1的列放在一起，并把它们分开作为按顺序排好形成新的两组。

如果只有一组内有列在该行取到1，我们把该组内所有在该行取值为1的列放在左边分成一组，把取值为0的列放在右边，叫做“特殊组”，因为这组可以取出任意列放在原来组内所有列的最左边并组成一个新的“普通组”。但是这样“特殊组”在以后可能会出现选择不唯一的情况：既可以取出来放在这些列的左边也可以移到该“特殊组”内的最左边。为了避免这种情况的发生，我们可以先加入1的个数最多的行，这样割出的“特殊组”是不会产生这种情况的。

考虑了所有行的情况后我们就可以得到一个合法解了。时间复杂度$O(N^3)$，也可以做到$O(N^2 \log N)$。用PQ树可以做到$O(N^2)$，不过非常复杂且应用比较狭窄。

13 Codeforces 248E Piglet's Birthday

题目简述

有N堆数，一开始第i堆数有$a_i(0 \leq a_i \leq 100)$个1。现有$Q(\leq 10^5)$次操作，每次在一堆内等概率随机选择$K(\leq 5)$个数，把它们变成0并放入另外一堆中。问每次修改后期望有多少堆数没有1。

算法分析

显然每次修改后1的个数（也就是没有被选出的个数）不会增加。

我们可以设$f_{i,j}$表示第i堆数当前有j个1的概率，答案就是$\sum f_{i,0}$。设某次修改从第u堆向第v堆移了k个0，那么$f_{v,0}$的概率不会改变，可能发生变化的只有$f_{u,0}$，所以我们每次重新计算f_u。转移方程比较显然，枚举这次取走了几个1，计算概率即可。

时间复杂度$O(Q * 100 * 5)$。

14 Codeforces 249D Donkey and Stars

题目简述

在笛卡尔座标系的第二象限上有$N(\leq 10^5)$个点，从点x可以移到点y的条件是从点y在点x射出的两条与x轴夹角为α_1和α_2的射线之间的区域。问从原点出发最多可以移动多少次。

算法分析

我们把一个点可以转移的区间画出来，然后试图把它和我们熟悉的模型建立起联系。

对于二维问题我们常考虑将一维排序后，另一维用某些结构来维护操作，比方说最长不下降子序列，它本质上也是如此。

注意到最长不下降子序列在二维平面上的转移其实和本题非常类似（夹角为0度和90度），所以本质上我们把座标旋转后就能转化为该问题了。具体来说分别把x座标旋转α_1度，y座标旋转α_2度后就和最长不下降子序列问题相同了。

时间复杂度$O(N \log N)$。

15 Codeforces 249E Endless Matrix

题目简述

有一个如下图所示的无穷矩阵，求其中一个子矩形的元素和，如果答案大于10位的话则先输出"..."再输出最后10位，子矩形座标不超过10^9，有10^5组数据。

1	2	5	10	17	26
4	3	6	11	18	27
9	8	7	12	19	28
16	15	14	13	20	29
25	24	23	22	21	30
36	35	34	33	32	31

算法分析

很显然我们可以把子矩形的和拆成四个左上角为$(1,1)$的矩形然后容斥就求出答案了。

求解原点为左上角的矩形就方便许多了。我们可以把它拆成一个包括左上角的正方形和剩下的一个矩形。正方形的元素就是从1到该正方形的面积的所有自然数的和，而剩下的矩形的每列（行）的差是等差序列，每列的和形成的序列又是等差序列，它们都是可以$O(1)$算出来的。由于我们是对10^{10}取模，我们必须要用64位无符号整数来处理所有运算，并且要小心处理取模和除法的问题。

现在还有一个问题：如何判断是否要输出"..."呢？一个比较简单的方法是，用浮点数再算一遍，得到大概的答案后就可以判断是否要输出"..."了。

对于每组数据所需的时间都是常数级的。

16 Codeforces 251D Two Sets

题目简述

有$N(\leq 10^5)$个数$(\leq 10^{18})$，现在要把它们分成两个不相交的子集S_1, S_2（可以为空），问S_1中的数的异或和加上S_2中的数的异或和最大是多少。

算法分析

对于位运算的题我们常常从高位向低位贪心来确定答案。本题也不例外。

对于某一位来说，如果有奇数个数该位为1，那么分成的两个集合中有且只有一个集合使得其异或和的该位为1，所以我们可以先不考虑这些位。

如果有偶数（并且非0）个数该位为1，那么我们可以分成两个集合，使得它们各自含有奇数个该位为1的数，这样该位在两个集合中的异或和中都为1了。

由此我们可以从高位向低位贪心：对于所有偶数位，我们可以列出一个异或方程，如果方程组有解则该位可以被加入，否则该位不能被加入。

在消元的过程中记录交换的顺序就可以求出方案了。消元时可以压位加速成$O(1)$。时间复杂度$O(N \log 10^{18})$。

17 Codeforces 251E Tree and Table

题目简述

有一个$2*N(\leq 10^5)$的表格和一棵有$2*N$个点的树，现在要把树上的点放在格子中，要求在树上相邻的节点在表格上的映射也要相邻，求方案数模10^9+7。

算法分析

先处理两个特殊情况：当N为1时答案为2；当所有点的度都不超过2时答案为$2N^2-2N+4$。前者不用证明；后者的证明也比较容易：对于一个大小为$2*k$的格子并且链的一端放置在左上角的方案数是k，枚举链的一端放置的位置就可以得到前面的式子了。

那么如果存在某个节点的度数是3呢？我们把它找出来并且把它当做树根。我们需要枚举它的各个儿子的排列情况（6种），如果位于该点下面的点也有两个儿子的话我们还需要考虑它们的排列情况。

枚举了这些情况后我们就可以设定状态了：不妨设F_x和$G_{x,y}$分别表示以x为根的子树并且x位于子表格的左上角时的方案数，以及x和y位于表格一端时它们的子树构成的方案数。G的转移比较显然就不说了；我们考虑比较复杂的F的转移。

如果在以x为根的子树内没有度数超过3的节点的话，方法数就是$size_x/2$，其中$size$表示子树的大小。否则我们考虑离x最近的一个点w。考虑从x到w之间的一段链，如果我们要把这段链横着放的话，我们就要枚举w的某个儿子（或者是w的某个儿子的某个儿子）来填上这段链下面的空隙，在这之后我们将转移到某个F或者某个G，具体画出来就可以知道了。如果我们不横着放这段链的话就从x的儿子的儿子转移过来就行了（如果可行的话）。

总的时间复杂度是$O(N)$的。

18 Codeforces 254D Rats

题目简述

在$N*M(N,M\leq 10^3)$的棋盘中，每个格子是要不是被老鼠占领，要不就是空地或障碍物。现在要投放两枚手榴弹把老鼠全部炸死，手榴弹可以炸掉与它在棋盘上的距离不超过$D(\leq 8)$的老鼠，求手榴弹投放的位置，手榴弹不能经过障碍物，也不能在障碍物上投放。

算法分析

任意选取一只老鼠。显然至少有一枚手榴弹的投放点在离它不超过D的格子中，在这之内枚举第一颗手榴弹的范围。然后我们再选取一只没有被炸到的老鼠，枚举它被放在哪里的手榴弹炸掉的，最后检查一遍是否所有的老鼠都被炸掉即可。

由于$D\leq 8$，一次BFS最多只会有141个格子被访问到，总的时间复杂度就是$O(141^3)$。

19 Codeforces 256D Liars and Serge

题目简述

有一个长度为$n(\leq 2^8)$的数组a，每个元素a_i的取值范围为$[1..n]$。对于每个a_i，如果a_i在a中出现的次数不为a_i，那么我们认为a_i是“不好的”。给定n和“不好的”a_i的个数k，求有多少个不同的a满足要求。答案对777777777取模。保证n是2的整数次幂。

算法分析

设$f_{i,j,k}$表示考虑前i个数，已经确定了j个元素的取值，其中k个元素的取值是“不好的”的情况下共有多少种方案。

转移时我们枚举第i个数在a种出现了p次，同时用组合数算出可能出现的位置的情况数，具体方程如下：

$$f_{i,j,k} = f_{i-1,j-k,k} * \binom{n-j-k}{k} + \sum_{p=0}^{min(j,k)} f_{i-1,j-p,k-p} * \binom{n-j-p}{p}$$

上式中$p \neq i$。

时间复杂度$O(n^2k^2)$，对于$n = 2^7$是没有问题的，$n = 2^8$时打表即可。

20 Codeforces 257E Greedy Elevator

题目简述

有$N(\leq 10^5)$个人准备坐一台电梯。它们会在某个时刻到达某个楼层，前往另外的某个楼层。电梯的走向是由电梯内的人的方向和正在等的人决定的，它会向人数较多的一侧移动。问每个人最终到达的时间。

算法分析

首先把所有人的到达时间离散化，然后顺次处理各个事件。

每次我们找出时刻最早的没有被处理的事件，这个可以用平衡树来维护。我们同时用另外一个平衡树（支持查询不小于某个数的元素的个数查询即可）来维护当前在等的是向上的较多，还是向下的较多。

通过这个结构我们还可以求出下一个离开电梯的人（或者将会搭上电梯的人）是谁和变化的时刻，并将该事件与下一个会发生的事件比较。如果该事件较前则优先处理该事件。

时间复杂度$O(N \log M)$。

21 Codeforces 260E Dividing Kingdom

题目简述

在无限大的笛卡尔座标系中有$N(\leq 10^5)$个点，现在要平行于x轴切两刀，平行于y轴切两刀，要求切出的9个部分内包含的点数为给定的值（或其排列）。

算法分析

我们可以枚举所有的排列，这样同时确定了切的方案，我们只需要检查每部分是否合法即可，也就是查询一个矩形内包含的点数。

查询二维平面内某个子矩形内的点数有很多办法。由于座标范围较大，我们需要先离散化处理。

可以直接平衡树套vector二分查询（$O(\log^2 N)$）或者按照一维排序后，用主席树来查询（$O(\log N)$）或者离线后将所有询问按照一维排序，另外一维用数据结构维护，扫描线解决（$O(\log N)$），总的复杂度就是$O((N+9!)\log^2 N)$或者$O((N+9!)\log N)$。

22 Codeforces 261D Maxim and Increasing Subsequence

题目简述

有一个长度为$N*T(N \leq 10^5, T \leq 10^9)$的正整数序列$a$，其中$a_i = b_{(i-1) \bmod N+1}(1 \leq i \leq N*T)$。设$maxB = max(b_i)$，那么有$1 \leq maxB \leq 10^5, N*maxB \leq 2*10^7$。求该序列的最长上升子序列。

算法分析

分析一下就会发现T是不会超过N或者$maxB$的。

考虑直接维护f_j表示当$a_i = j$时前i个数的最大值，显然f_j是不减的。又注意到所有f_j的总增量是不超过$N*T$的（也就是小于$2*10^7$），我们直接暴力维护即可。

具体来说，我们顺次处理每个a_i，并且维护加入a_i后f的变化，显然是只会在$j \geq a_i$的f_j中发生变化。我们判断f_{a_i}是否需要更新，如果更新的话，我们则再次维护f的单调性。

时间复杂度$O(N*T)$。

23 Codeforces 261E Maxim and Calculator

题目简述

有a, b两个数，一开始$a = 1, b = 0$。每次操作有两种选择：把a赋值为$a * b$，或者把b加上1。问在区间$[l, r](l \leq r \leq 10^9)$内有多少个数$x$可以在是操作不超过$p(\leq 100)$步后使得$a = x$。

算法分析

由于$b \leq p$，可以在a中出现的数的质因数一定不会超过p。搜索后可以发现在10^9之内只有不到300万个数符合要求。我们搜出所有$\leq R$的合法的数，之后的事情就是DP了。

设$f_{i,j}$表示b最多为i时最少需要多少次$a = a * b$的操作使得$a = j$。那么方程也很显然了：$f_{i,j} = min(f_{i-1,j}, f_{i,j/i})$。

时间复杂度$O(pn)$，n是状态数。

24 Codeforces 263E Rhombus

题目简述

给定一个$N * M(N, M \leq 1000)$的矩阵a和$K(1 \leq K \leq \frac{min(N,M)}{2})$，求使$\sum_{i=1}{}^{N} \sum_{j-1}^{M} a_{i,j} * max(0, K - |i - x| - |j - y|)$最大的$x, y(K \leq x \leq N - K + 1, K \leq y \leq M - k + 1)$。

算法分析

设$s_{i,j}$表示以$(1, 1)$和(i, j)为对角内的元素和，我们可以维护两种对角线上的$s_{i,j}$的和，这样可以模拟出距离矩阵中的锯齿。

通过简单的容斥后就可以在$O(1)$时间内算出题目中要求的式子，直接更新答案即可。时间复杂度$O(N * M)$。

25 Codeforces 264D Colorful Stones

题目简述

有两个长度不超过10^6的仅由RGB组成的串S和T以及两个指针，一开始指针指在两串串首。每次两个指针会读取一个字符，如果与其所在位置匹配则前进一步，否则留在原地。问共能到达多少对不同的位置。

算法分析

容易知道，当一个指针的位置确定时，另一个指针的位置的两端都是单调的。

两端的维护很简单：对于右端来说，只要不能匹配则都可以前进。但在这段区间中，有一些位置是不可能停下来的（相邻两个的值和最右端以及下一个的取值相同且两个位置不同），我们把这些值减去即可。

时间复杂度$O(|S| + |T|)$。

26 Codeforces 264E Roadside Trees

题目简述

在一条路上从西到东有$N(\leq 10^5)$个位置，在接下来的$M(\leq 2*10^5)$分钟里都会发生以下事件中的一种：

1. 在位置p上种一棵高度为$h(1 \leq h \leq 10)$的树。
2. 砍掉从左到右数的第$x(1 \leq x \leq 10)$棵树，并且该位置以后也不会种树。

在每分钟每棵树都会变高1单位，现在询问每分钟结束时，树的高度构成的最长上升序列是多少，保证在任意时刻没有两棵高度相同的树。

算法分析

对于每棵树在每分钟都长高的这个条件，简单地把树高减去时刻后就可以忽略了。

不妨维护从每棵树出发可以取到的最长上升序列的长度f。题目中有两个非常重要的信息：新种的树的高度不超过10，而砍树的话只会砍前10棵树。

首先考虑种树。由于新种的树的高度不超过10且在任意时刻没有两棵高度相同的树，而树的高度又是正整数，那就是说比新插入的树矮的树最多只有9棵。注意到只有比新树矮的树的f值才可能发生变化，我们可以重新算出这10棵树的f值，只需要维护一个按照位置建的线段树就可以了。

砍树的情况类似：因为只会砍前10棵树，只有排在被砍树之前的树的f值才可能发生变化。相似地，我们也只需要重新算出这些树的f值，这时我们需要维护的线段树是关于高度建的。更新的时候两个线段树要同步更新。

总的时间复杂度是$O(M*10\log M)$。

27 Codeforces 266D BerDonalds

题目简述

给定一个$N(\leq 200)$个点的无向带权联通图，求一个中心，使得该处到最远的点的距离最小。

算法分析

首先求出任意两点间的最短路，接下来枚举中心落在哪条边上。

那么对于这条边(u,v)，从该边上出发到达某个点，一定会存在某个分界点，使得一边的最短路是经过u，而另外一边的最短路经过v。我们可以按照u到另外的点的距离对其他点排序。对于从v离开的路径，我们可以维护一个前缀最大值，来求出在此时要经过v离开的距离的最大值是多少。

时间复杂度在预处理排序值后可以做到$O(N^3)$。

28 Codeforces 267C Berland Traffic

题目简述

一个$N(\leq 100)$个点$M(\leq 5000)$条边的无向图，每条边有每单位时间内能通过的流量的限制，除了1号点和N号点都要满足流量平衡，同时要满足对于任意一条从x到y的路径上的流量和相同。求在一单位时间内从1到N和从N到1的流量和最大是多少并求出具体方案。

算法分析

先不考虑每条边具体一时刻内流过多少流量，我们先求出每条边流过的流量站总流量的比率是多少，最后乘上一个倍率就好了。又由于我们限制了两点之间的流量是一定的，我们不妨用点经过的比例来列方程，这样我们可以保证两点经过的流量与具体经过的边无关。那么对于某条边来说经过的倍率就是两点之间倍率的差。

时间复杂度$O(N^3)$。

29 Codeforces 268D Wall Bars

题目简述

有一个长度为$N(\leq 1000)$的仅由1234构成的序列。设一条路径是从前$H(\leq 30)$个字符开始的相邻位置不超过H的同色位置的序列。对于一个合法的序列，要求至少有一条到达位置$N-H+1...N$的路径。求合法序列数。

算法分析

设$f_{i,j,k,l,u}$表示考虑前i个位置时，另外三种颜色离i的距离分别为j,k,l，第i个位置的可达性为u时的方案数。

转移比较明显，枚举位置i来自哪种颜色即可。顺推状态会比较好实现，更新在新增位置i后j,k,l以及u值就可以完成转移。

时间复杂度$O(NH^3)$。

30 Codeforces 269D Maximum Waterfall

题目简述

有$N+2(\leq 10^5)$个平台，其中一个在高度0, 一个在高度T。每个平台有垂直位置（高度）和水平位置（一段区间）。现在有水从高度为T的平台流向高度为0的平台，水只会由高向低流，如果两个平台的水平位置有重叠，而且没有一个高度在它们之间的平台和它们的位置都有重叠的话，它们之间就可以流两者的水平重叠长度那么多个单位的流量。问一条水流可能的最大流量是多少。

算法分析

如果a能流到b的话，我们就从a向b连一条单向边。显然这个图是平面图，它的边数是$O(N)$级别的。

考虑如何构造出这个平面图。我们按照高度排序，用平衡树来维护当前平台覆盖的情况，那么对于某个平台来说，连向它的边就是覆盖了该平台水平位置的所有平台，在平衡树上直接二分查找就可以了。

构造出了这个图上的所有边后，我们还要判断是否有一个高度在它们之间的平台和它们的水平位置都重叠。具体来说，只用判断与该平台在平衡树中相邻的平台是否会干扰就可以了。

时间复杂度$O(N\log N)$。

31 Codeforces 269E String Theory

题目简述

在一个$N * M(N, M \leq 10^5)$的矩形的边上均匀地分布着$2N + 2M$个洞，有$N + M$条弦把洞配对，现在可以交换任意两行或者两列的洞，要求最后所有的弦不相交。

算法分析

首先我们把弦分类：有左上，右上，左下，右下，左右和上下六种不同的弦，显然左右和上下是不能同时出现的，否则肯定没解。为了描述方便不妨设只有上下的弦。

第二个注意的是不管怎么交换行列弦的分类是不会改变的，这可以告诉我们，对于一个给定的输入，最后构成的形态是固定的。

我们不妨把某个位置和它相对的位置连上边，注意到这样构成的环在交换行列的过程中是不会改变的。进一步的观察我们可以发现，只会有两种不同的环，一类依次出现在矩形的四个角，长度为4；另外一类环的长度是固定的且循环同构。前者的处理比较显然，后者要用KMP或者最小表示法来处理循环同构的问题，这样我们就可以生成一个合法的排列了。

时间复杂度是线性的。

32 Codeforces 273D Dima and Figure

题目简述

在$N * M(N, M \leq 150)$的矩形中计算有多少个联通块，使得其中任意两点在联通块中的距离恰为其曼哈顿距离。

算法分析

题目中要求的东西就是凸联通块的个数。假设我们从左到右地去看这个联通块，它的上边界和下边界都是凸的。对于上边界来说它先是一段不减的序列，接着一段不增的序列。下边界恰好相反。

这样我们就可以设状态了：设$f_{i,j,k,s}$表示第i列的下边界在j，上边界在k，当前边界状态为s时的方案数。我们一共有四种状态：上下边界都未转换状态，仅上边界转换了，仅下边界转化了以及上下边界均转化了，其中仅上边界转化和仅下边界转化是类似的，可以记为同一种状态，最后把答案乘2就可以了。

考虑转移。我们可以维护二维下的前缀和（求子矩阵）就可以在$O(1)$的时间内维护出f了。时间复杂度$O(NM^2)$。

33 Codeforces 273E Dima and Game

题目简述

有$N(\leq 1000)$个区间$[l_i, r_i](1 \leq l_i < r_i \leq P)$，两个玩家依次操作。在一次操作中每个玩家可以选择一个$r_i - l_i > 2$的区间，并将它变成$[l_i, r_i - \lfloor \frac{r_i - l_i}{3} \rfloor]$或者$[l_i + \lfloor \frac{r_i - l_i}{3} \rfloor, l_i + 2 * \lfloor \frac{r_i - l_i}{3} \rfloor]$，最后一个操作的玩家胜利。给定$N, P$，假设双方都采取最优策略，有多少个使先手获胜的序列。

算法分析

首先我们可以发现每个区间只和区间的大小有关。这个问题显然是N个游戏的和，我们来分析单个游戏的sg值的分布。

由于只有两种转移，sg函数的取值必定在2以内。打表观察发现该函数的分布可以分成若干段，其中每段中的sg值都相同。进一步计算我们可以知道段数是$\log P$级别的，这样我们可以在$\log P$的时间内算出所有的sg值了。

算出所有sg值后就是DP计数了，这部分比较简单就不叙述了。时间复杂度$O(N + \log P)$。

34 Codeforces 274C The Last Hole!

题目简述

平面上有$N(\leq 100)$个圆，开始半径均为0，在时刻t所有圆的半径均为t。问从哪个时刻开始，之后圆之间不会再产生洞。

算法分析

可能成为最后的洞的地方只可能是锐角三角形的外心或矩形的中心。换句话说，钝角和直角三角形不存在洞。

这个性质的证明比较直观，因为外心根本就不在三角形内，当然不可能构成闭合的洞。

算出外心后还需要检查该点是否会被其他点覆盖。

时间复杂度$O(N^4)$。

35 Codeforces 274E Mirror Room

题目简述

在一个$N * M$的网格上有K个格子被镜面障碍物占据了($N, M, K \leq 10^5$)。现在你在某个空格以某方向（左上，右上，左下和右下）发射一束激光，激光撞上障碍物或边界时会镜面反射，问共有多少个格子被光穿过了。

算法分析

显然在一段时间后激光的轨迹形成了一个环。找出这个环的构造之后就可以构造出问题的解了。但是有几个问题：是否有格子在环上重复出现，和这个环是否一定回到起点。

考虑第一个问题。我们把格子黑白染色，如果发生了一次镜面反射，那么经过的格子的颜色就会得到变换，同时光路旋转了90度，而又由于在发生反射前光路所经过的格子的颜色都是相同的，我们可以知道，一个格子只可能被一对对角连成的直线访问到，不是左上——右下方向上的，就是左下——右上方向上的，因此一个格子只会被访问一次。

对于第二个问题，我们可以看到，一旦起点和方向确定，对于每个格子的前驱和后继都是确定的，因此整个路径时一个完整的环，起点也在其中。

剩下的事情就是模拟光路了。判断前方的障碍物可以用平衡树来完成，总的反射数是$O(N + M + K)$级别的，时间复杂度$O((N + M + K) \log K)$。

36 Codeforces 277D Google Code Jam

题目简述

有$N(\leq 1000)$道题，每题分小数据和大数据，各自需要花一定时间来完成，正确回答后得到一定的分数，小数据一定能做对，大数据有一定的概率会做错，最后一个正确解的完成时间称为罚时。问在最优策略下$T(\leq 1560)$分钟内最高的期望得分和在此前提下最小的期望罚时。

算法分析

容易得知我们肯定会先完成小数据再做大数据，这样肯定比先做某个大数据再做小数据优。

显然小数据之间的顺序是随意的，但是大数据之间就不是这样了。设完成大数据的时间为t，错误概率为p，那么大数据之间的顺序就是按照$t*p/(1-p)$从小到大排序得到的序，证明也很简单：设有两题所需时间分别为t_1,t_2，错误概率分别为p_1,p_2，那么先做前一题的期望罚时是$p_2*(1-p_1)*t_1+(1-p_2)*(t_1+t_2)$，先做后一题的期望罚时是$p_1*(1-p_2)*t_2+(1-p_1)*(t_1+t_2)$，化简后就得到上式。

得到序之后就是DP了。设$f_{i,j}$表示前i题花了j分钟的最高期望得分，$g_{i,j}$表示在$f_{i,j}$取到最优时的最少期望时间。转移类似背包。时间复杂度$O(N*T)$，可能要用long double来解决精度问题。

37 Codeforces 280D k-Maximum Subsequence Sum

题目简述

有一段长度为$N(\leq 10^5)$的序列，现在有$M(\leq 10^5)$个操作，每个操作会是以下操作中的一种：

1. 将某个位置的数修改
2. 在l到r之间选取不超过$k(\leq 20)$段不相交的子序列，求这些子序列的和的最大值

算法分析

如果只取一段的话，我们显然可以用线段树维护最大值来解决问题。

要取多段的话，我们不妨运用流的思想：我们允许重复选元素，某个元素选了的话就把它取反，这样再选到它的时候就相当于取反了。我们再做最多k次查询，每次把查询到的最值区间取反。正确性显然和流的正确性是相同的，每次也只会增加一段实际上选的区间。

时间复杂度$O(MK \log N)$。

38 Codeforces 280E Sequence Transformation

题目简述

有一个长度为$N(\leq 3*10^5)$的不减序列a，现在要你把a转换成另外一个序列y，使得$1 \leq y_i \leq q$且$a \leq y_{i+1} - y_i \leq b(0la \leq b)$，并且转换代价$\sum i = 1N(a_i - y_i)^2$最小。

算法分析

设$dp_{i,j}$表示第$y_i = j$时前i个数的最小花费是多少，转移很显然就不打了，但直接这样做是不行的，因为y_i的值域是实数，我们有无限个状态，直接设是不行的。

单独求解不得，我们不妨考虑整体求解：把dp_i当成一个函数，求出它的表达式进而知道具体的$dp_{i,j}$的值，这样做是因为容易得到dp_i是连续且是严格单峰的，考虑表达式的一阶导数的变化就可以知道了。

知道dp_i是单峰的话我们的转移就可以这样写了：找到dp_{i-1}的极值，把该位置左边整体右移a，右边整体右移b，这样中间就空出一段长$b-a$的平台了，最后整体加上函数$(x-a_i)^2$即可。

这些操作的维护都可以用平衡树打标记来完成，总的时间复杂度就是$O(N \log N)$。

39 Codeforces 283E Cow Tennis Tournament

题目简述

有$N(\leq 10^5)$头牛，每头牛有一个唯一的能力值，能力值较高的牛获胜。现在有$Q(\leq 10^5)$次操作，每次将双方能力值都在一段区间内的牛的比赛结果翻转，问最后有多少对无序三元组(u, v, w)满足u战胜v，v战胜w，w战胜u。

算法分析

直接算合法三元组比较困难，正难则反，我们考虑计算不合法的三元组的数量。

在一个不合法的三元组内，必定有且仅有一头牛战胜了另外两头牛。换句话说，我们只要统计出每头牛战胜了多少头牛就可以算出有多少个不合法的三元组了。

我们把能力值排序后，按序扫描翻转事件，用线段树维护当前牛和其他牛的胜负关系，区间打标记和求和即可。

时间复杂度$O((N + Q) \log N)$。

40 Codeforces 285E Positions in Permutations

题目简述

在一个长度为$N(\leq 1000)$的排列p中，$|p_i - i| = 1$的位置被认为是完美的。问长度为N的恰有K个完美位置的排列有多少个。

算法分析

设f_i表示至少有i个完美位置的排列有多少个，那么最终答案容斥就可以算出来了。

设g_i表示考虑这i个完美位置之内的排列有多少个。假设某个$p_i = i + 1$，且$p_{i-1} \neq i$，那么如果要让$i + 1$成为完美位置只有两种选择，如果$p_{i+1} = i + 2$的话，那么位置i就不可能成为完美位置了。

我们枚举两种放法的个数，用组合数就可以算出g_i和f_i了。

时间复杂度$O(N^2)$。

41 Codeforces 286D Tourists

题目简述

有$N(\leq 10^5)$条线段，第i条线段自时刻t_i其出现在$(0,l_i)$与$(0,r_i)$之间的连线上。现在有$M(\leq 10^5)$个询问，每次询问从某个时刻开始，在$(0,0)$与$(0,\infty)$之间被线段覆盖的长度。

算法分析

我们把座标离散化，把线段按照插入时间排序，那我们需要做的就是统计每段区间内最早被哪条线段所覆盖。

如果按照时间顺序处理的话，我们可以保证每个区间内只被时间最早的线段覆盖。这要求我们找到下一个没有被覆盖的区间，显然可以用并查集来维护。

我们再把每次插入的增量存储下来，在询问时刻增加时加上增量就可以了。

时间复杂度$O(N\log N)$。

42 Codeforces 286E Ladies' Shop

题目简述

有$N(\leq 10^6)$个正整数a_i，现在你需要找到一个集合$p_1, p_2, ..., p_k(1 \leq p_1 < p_2 < p_k)$并满足以下条件：

1. 对于所有a_i都需要存在$y_1, y_2, ..., y_k(y_i \in \mathbb{N})$使得$\Sigma p_j * y_j = a_i$
2. 对于所有在$[1..M(\leq 10^6)]$内的不等于任意一个a_i的正整数x都不存在$y_1, y_2, ..., y_k(y_i \in \mathbb{N})$使得$\Sigma p_j * y_j = x$
3. 在满足以上条件的同时最小化k

算法分析

由第二个条件告诉我们，集合p中的元素都是a中的元素。

为了减少元素的数量，我们只用选取那些不能被其他数所表示的数。显然如果有解的话这样的选法一定是最优的。

那么怎么统计该数是否能被其他数表示呢？我们显然可以做一个DP，统计有多少种不同的方法能够取到i，当$f_i = 1$时就是要加入答案中的数。这样顺便判断了无解的情况。

但这样是$O(N * M)$的，我们需要优化。

注意到我们关心的其实只是f_i是否大于0和是否大于1，对于第一个会导致无解的位置，它一定是由两个a中的元素加起来所导致的；而如果有解的话，如果某个数能被其他数表示，那么它一定能被两个a中的元素表示出来。

所以我们就可以用FFT了。直接用复数运算是可以保证精度的。

时间复杂度$O(M \log M)$。

43 Codeforces 288E Polo the Penguin and Lucky Numbers

题目简述

如果某个正整数只包含4或7的话，它就被认为是幸运数。给定两个幸运数$l, r(l < r < 10^{10^5})$且它们的位数相同，将l到r之间的幸运数从小到大排在一行，求相邻两数的乘积的和对$10^9 + 7$取模。

算法分析

显然答案可以表示成前缀和的差。

考虑从高位到低位进行操作。在增加一位的时候所有原来的幸运数都被加上一位，而加上4或7后它们的增量都是通过直接求和算出来的。同时我们需要维护当前的最小值和最大值，这样才能在增加位数时快速算出答案。

时间复杂度$O(\log_{10} r)$。

44 Codeforces 293B Distinct Paths

题目简述

在一个$N * M(N, M \leq 10^3)$的棋盘上，有一些格子已经被涂上$K(\leq 10)$种颜色中的一种。问有多少种涂色的方法使得从左上角到右下角的每条路径都不会经过两个颜色一样的格子，路径只能向右或向下走。

算法分析

$N, M \leq 10^3$纯属唬人，当$N + M - 1 > K$时显然就无解了。

考虑直接搜索。注意到我们只需要考虑“本质不同”的颜色。考虑对称性。假设当前搜索的格子要涂一种还没有涂过的颜色，即还没有搜索过并且没有在读入时就用过的颜色，这个格子涂任何一种这样的颜色对后续搜索的影响都是一样的，此情况只用搜索一遍。

同时我们考虑在每一步搜索时都判断合法性，我们只需要记录上该颜色上次出现的位置就可以$O(1)$判断了。

这样的搜索量得以大大减少，即使棋盘非常稀疏也能瞬间出解。

45 Codeforces 293D Ksusha and Square

题目简述

给一个$N(\leq 10^5)$个顶点的凸多边形，任意选定两个在凸多边形内（包括边界上）的点，求以它们的连线为对角线的正方形的面积的期望。座标的绝对值$\leq 10^6$。

算法分析

我们实际上的就是要求$\Sigma(x_i - x_j)^2 + (y_i - y_j)^2$。从式子的形式来看我们可以对$x$座标和$y$座标分开求和。

由于座标的范围比较小，我们可以求出在某一维确定时另外一维的取值范围。把上面的式子化简之后就是对二次项和一次项分开求和，然后就可以算出答案了。时间复杂度是关于座标范围线性的。

46 Codeforces 293E Close Vertices

题目简述

给一棵有$N(\leq 10^5)$个节点的树，每条边有权值，问有多少个点对满足两点之间的简单路径长度$\leq L$，边权和$\leq W$。

算法分析

本题就是树中点对距离的二维关键字版。

仍然是用点剖来解决问题。第一个关键字我们可以通过排序后单调地扫描来确定查询的范围。

而第二个关键字我们可以在当前查询范围内用树状数组维护前缀和，在查询区间变化时单点修改时即可。

为了保证查询的点对的LCA为当前点剖的根，我们需要减去根的每棵子树内部的值。

时间复杂度仍是$O(N \log^2 N)$。

47 Codeforces 294D Shaass and Painter Robot

题目简述

给定一个$N * M(N, M \leq 10^5)$的棋盘，从某一点向某方向发射一束光标，将经过的格子染色，碰到边界则镜面反射，问把棋盘黑白染色后所经过的格子数，同一个格子经过多次时会被计算多次，或者返回无解。

算法分析

我们考虑直接模拟走的过程。

本题一定会构成一个回到原点的环，因为前继和后继都是唯一确定并且存在的，这样最后形成的显然是一个（或若干个）环。

本题由于不存在障碍物，我们可以直接确定它下一次碰撞的位置（一定是边界上的某个位置）和增加的步数，按照镜面反射的规律来。

当我们经过了$N + M - 2$个不同的边界上的点时就找出了解，这也是判断是否完成的充要条件。

时间复杂度$O(N + M)$。

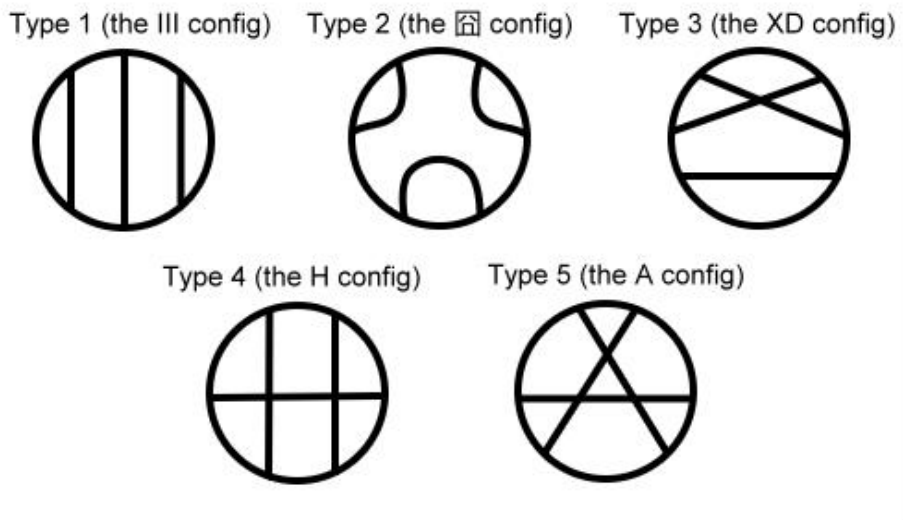
48 Codeforces 297E Mystic Carvings

题目简述

在一个环上有$2 * N(\leq 100000)$个点，有N条线段把这些点匹配，要求选三条边使得三条边端点之间的距离相同，其中距离的定义是两点之间在环上最少要经过多少个被选过的边的端点。求方案数。

算法分析

选三条边的话共有5种情况，如图所示：



那么我们要统计的就是第二种（“囧”）和第五种的个数。不幸的是它们都比较难求。不过也有好消息，另外三种都比较好求。

设in_i表示被第i条线段包含的线段个数，那么与线段i相交的线段个数$cross_i$就是$r_i - l_i - in_i - 1$，求in可以按顺序扫用树状数组维护前缀和即可。

对于第一种我们枚举中间的那条线段i，它的贡献就是$in_i * (N - 1 - in_i - cross_i)$。而对于三和四都是对于一条线段，有一条相交，一条不相交。我们枚举那条线段，贡献就是$in_i * (N - 1 - in_i)$，最后再除以2（算重），就算出不合法的方案数了。

时间复杂度$O(N \log N)$。

49 Codeforces 301C Yaroslav and Algorithm

题目简述

设计一个如下的算法：

1. 这个算法接受一个字符串作为输入。我们设这个输入字符串为a。
2. 这个算法由一些命令组成。i号命令的形式为"$s_i >> w_i$"或"$s_i <> w_i$"，其中s_i和w_i是长度不超过7的字符串（可以为空），由数字或字符"?"组成。
3. 这个算法每次寻找一个编号最小的命令i，使得s_i是a的子串。如果没有找到这样的命令，那么整个算法终止。
4. 设找到的命令编号为k。在字符串a中，s_k第一次出现的位置会被w_k替换。如果这个命令形如"$s_k >> w_k$"，那么这个算法继续执行（回到第3步）。否则，算法终止。
5. 算法的输出就是算法终止时字符串a的值。

并且该算法的返回值为输入串对应的十进制数加一后的字符串。

算法分析

考虑直接构造通用算法。加一的本质就是将结尾的若干个9变成0，再将前一位的数加1。为了判断当前操作的位置，我们可以用问号来标记。

相似地，我们需要在最后一个非9的整数后打标记。此时出现了不同的标记，我们不妨将一个问号变成两个问号，这样两个问号代表进位还未完成。我们最后再给每个数进行加法操作即可。

一共需要23条命令来处理打标记以及标记的进位操作。操作步数约是$2k$次，其中k是位数。

50 Codeforces 301E Yaroslav and Arrangements

题目简述

定义一个长度为N的数列a为好的，当且仅当相邻以及头尾两数的绝对值差为1，且a_1为数列中的最小值。定义数列b为优秀的，当且仅当数列是不降的，长度不超过N，最大值不超过M，且重排数列后可以得到至少一个至多K个好的数列，其中两个数列被认为不同仅当存在某个位置其对应值不同。给定$N, M, K(\leq 100)$，求优秀数列的个数，对10^9+7取模。

算法分析

这类计数类的题目比较容易想到用DP来解决。

由于头尾两数的差为1且第一个数是最小的，我们把数列查分后容易看出它是一个括号序列且是完美匹配的。这样我们就可以把问题的规模减半了。设$dp_{i,j,k,p}$表示已经用了i对括号，上一个数用了j次，目前用了k种数，不同的数列的方法数为p时的方案数。转移就比较暴力了：我们枚举第$k+1$个数出现了几次，为了不重复统计我们只能在k后面插入$k+1$，设$k+1$有q个，那么方法数就是C_{j+q-1}^{q-1}，转移到$dp_{i+q,q,k+1,p*C_{j+q-1}^{q-1}}$即可（如果这个状态是合法的话）。

看起来是五次方级别的，但是我们要对生活抱有美好的期待：实际上有用的状态是很少的！即使N, M, K均为100也是可以瞬间出解的。

51 Codeforces 303D Rotatable Number

题目简述

定义B进制下的“可旋转数”为：在B进制下该数乘1到它的长度乘出来的数均可以由其最后若干位数字移到最前面构造而来。给定N和X，找到最大的一个$B(1 < B < X)$，满足在b进制下存在一个长度为n的正“可旋转数”（允许有前导零）。

算法分析

这是一个结论题。

由Wikipedia可知，“可旋转数”的形式为$\frac{b^{(p-1)}-1}{p}$，p为质数，$(b,p) = 1$，且b为模p的原根。又由于$N = p - 1$，我们判断$N + 1$是否为质数，是的话则从b大到小判断是否为模p下的原根即可。

具体的判断是判断每个$b^{\frac{p-1}{i}}$模p是否都为1，所以判断一个数的时间复杂度是$\sqrt{N}\log N$的。

52 Codeforces 303E Random Ranking

题目简述

有$N(N \leq 80)$个人参加一场比赛，第i个人的得分是$[l_i, r_i]$之间的随机的一个实数，求每个人排在各种名次的概率。

算法分析

首先把区间离散化。设$f_{i,j,k,p,q}$表示第i个人的得分落在第j个区间，考虑前k个人时有p个人的得分在j之前，有q个人的得分在第j个区间内时的概率。转移也非常显然。最后假设有q个人和i落在j中，那么他们之间的排名的概率是相等的。时间复杂度是$O(N^5)$，勉强可以碾过去。

注意到如果确定了j，对于每个不同的人i，它们的转移是非常相似的。由此我们可以用分治来处理前缀的答案和后缀的答案，最后把它们合并起来，时间复杂度$O(N^4 \log N)$。

53 Codeforces 306C White, Black and White Again

题目简述

有W件不同的好事和$B(W, B \leq 4000)$件不同的坏事，在接下来的$N(\leq 4000, W + B \geq N)$天中每天至少会发生一件事情且每天只会发生好事或坏事。问有多少种不同的方案使得这n天会先有若干(¿0)天发生好事，再有若干天发生坏事，再有若干天发生好事，一天中发生的事的顺序不同也算不同的方案。对$10^9 + 9$取模。

算法分析

由于每天内发生的顺序不同也算不同，我们可以预先排列好好事之间的顺序和坏事之间的顺序，这样一共有$W! * B!$种方案。

现在要把B件坏事插在W件好事中，我们可以枚举插入的位置($W -$ 1种)，再枚举另外$N - 3$个空插在了哪些地方（有两个已经确定了），所以方案数为$W! * B! * (W - 1) * C_{W+B-3}^{N-3}$，组合数可以用逆元来处理。

时间复杂度$O(W + B + N)$。

54 Codeforces 306D Polygon

题目简述

构造一个$N(\leq 100)$边凸多边形使得每条边的长度都不同并且每个角都相同。

算法分析

首先$N \leq 4$时是无解的，在N较大时显然受到的约束较小。

我们考虑模拟出一个比较接近正N边形的图形。我们可以顺次减少每条边的长度，每次减少一点点（相对于初始长度较小，约10^{-5}级别）的长度，这样最后出现接不上的概率就非常小了。

为了保证最后几条边要结成一个完整的多边形，我们需要手工算出最后三个点转出的图形并判断可行性，如果不合法则要调整随机参数。但是出解的概率是非常高的。

一次随机的时间复杂度是$O(N)$。

55 Codeforces 309D Tennis Rackets

题目简述

一个等边三角形的每条边都被$N(\leq 32000)$个洞均匀地分成了$N+1$份，其中离顶点的距离不超过M单位的洞是无效的。问从每条边上选取一个洞组成一个钝角三角形有多少种取法。

算法分析

考虑枚举三角形的一个锐角所在的点和钝角所在的点，剩下的锐角可以取的范围是可以单调地维护出来的（另外一端就是取到序列的尽头）。

判断范围可以用余弦定理，判断三个点是否形成钝角的运算都是整数的。

这题主要的难度是正确的实现上面所述的方法，因为时限卡的比较紧。时间复杂度同为$O(N^2)$的算法最后跑出来的时间可以千差万别。

56 Codeforces 309E Sheep

题目简述

在直线上有$N(1 \leq N \leq 2000)$个区间，两个区间之间相连当且仅当它们之间有交集。求一个排列，使得相交区间对应的点在排列中的最大距离最小。

算法分析

显然答案是满足二分性的，不妨二分答案mid。

我们依次安排第i个区间是哪一个，设lim_i表示第i个区间当前受到的约束最前是哪里，那么一开始我们应该选择右端点最小的区间，这样增加的约束数不会比其他选法差。

每当我们确定了一个区间的位置后，lim就会被更新。更新后我们计算剩下的前i个位置要安排多少个区间，如果不够的话就返回无解。

那么下一个区间怎么选呢？此时再简单的选右端点最小的区间就是不对的了，因为此时有些约束会影响别的区间使得它们需求的位置更前。这时我们应该找出第一个“紧”的位置，具体来说就是有i个区间一定要放在剩下的前i个位置的话，那么这个位置就是“紧”的，在这些“紧”的区间中我们再选择一个右端点最小的区间即可。

时间复杂度$O(N^2 \log N)$。

57 Codeforces 311C Fetch the Treasure

题目简述

有$N(\leq 10^5)$个二元组(x,y)和一个集合S，初始时集合内仅有元素$K(\leq 10^4)$。如果某二元组在某时刻时满足$x = \sum_{i=1}^{|S|} S_i * k_i$，则将该二元组加入$Q$中。现在有$M(\leq 10^5)$个操作，每个是以下三个操作中的一种：

1. 在S中加入$a(\leq 10^{18})$
2. 减少某个二元组的y值
3. 询问当前Q中y值最大的二元组并将其删除

其中操作1的个数不超过20个。

算法分析

如果只有操作2和3的话，用堆就可以轻松地完成了，现在就只需考虑操作1的问题了。

对于所有的x，我们都把它们放在模K的同余系下考虑。对于每个$b(0 \leq b < K)$，我们都想知道最小的c使得$x = c * K + b$是可以被取到的。这可以在每次往S中加入a时重新做最短路解决。

时间复杂度$O(20K \log K + M \log N)$。

58 Codeforces 311E Biologist

题目简述

有$N(\leq 10^4)$个01变量和$M(\leq 2000)$条约束，每个变量有初始值和改变该变量所需的费用，第i个约束要求$k_i(\leq 10)$个变量均为一个指定值，若达到要求的话则获得收益w_i元。求最大收益。

算法分析

考虑把每个变量和每个约束都抽象成事件。假如只有当变量为0时才有收益的话，我们容易按照最大权闭合子图的思想建出图：原点向每个约束连流量为收益的边，约束向变量连流量为无穷大的边，变量如果初始值为1的话则向汇点连流量为改变它所需费用。

但变量为1时在本题中也有影响。怎么解决呢？反向套一个类似的网络即可，因为把整个图反向的最小割和最大流是不变的。

时间复杂度就是网络流的复杂度。

59 Codeforces 314E Sereja and Squares

题目简述

有一个只包含小写字母和问号的长度为$N(\leq 10^5)$的串，现在要在问号处填上小写字母或大写字母，使得所有字母可以被分成若干对，座标较小的是小写字母，座标较大的是对应的大写字母，求方案数。

算法分析

就是求括号序列的个数。直接暴力$O(N^2)$做DP即可，把上界和下界卡死一些，常数写小一些就可以过了，和309D类似。

60 Codeforces 316G Good Substrings

题目简述

设字符串t符合规则$(p,l,r)(0 \leq l \leq r \leq |p| \leq 50000,)$，当且仅当字符串$t$在字符串$p$中出现的次数在$[l..r]$内。给定$N(\leq 10)$条规则，问字符串$s(|s| \leq 50000)$中有多少个不同的子串符合所有的规则，其中两个子串不同当且仅当两个串中有一个位置不相同。

算法分析

考虑把所有串放在一起建后缀自动机。插入某个规则的字符时我们在最后插入的节点处打上标记。我们需要维护其fail树上某个节点子树内有多少个来自某个规则的标记，它代表的是该节点代表的子串在该规则内出现了几次。

时间复杂度$O(N * State + |\omega| * State)$，其中$State$代表后缀自动机的状态数，也就是字符数总和。

61 Codeforces 317E Princess and Her Shadow

题目简述

二维平面上有M个格子被树占据了。在每个时刻，公主会前往某个相邻且未被树占据的格子。与此同时，如果她的影子可以按照对应方向前进一步的话，它就会往该方向前进一步，否则会呆在原地。给定公主和她的影子的初始位置，构造一条路径使得公主和它的影子最后的位置相同，或者判断无解。

算法分析

无解的情况很简单，如果公主和影子之间不联通的话，那么显然是无解的。除此之外问题都是有解的。

假设只有一棵树，而公主在影子的左上方。一个比较简单的构造方法是，先把影子赶到树的上方，然后不断往下走直到在竖直方向上它们的位置相同，然后再把影子赶到树的左边，不断朝右走直到位置相同为止。对于其他的位置关系来说，它们的处理也是相似的。

对于多棵树的情况，如果我们有足够的空间完成上面的操作，那么问题就可以用上面的方法解决，否则意味着公主和影子都被树给围在里面了。这时我们找出公主和影子之间的最短路，让公主沿着这条最短路去追赶影子，同时记录影子按照最短路的方向行进时沿途的方向（不一定相同因为可能碰树）。显然公主和影子之间的距离是不减的，而又由于它们在同一个闭合的空间内，这样迭代的步数是有限的，我们一定能在有限步数内构造出解来。

62 Codeforces 319D Have You Ever Heard About the Word?

题目简述

有一个字符串$S(|S| \leq 50000)$，每次你需要找到最短的形如XX（可以在中间被分成相同的两份）的子串，有多个最短的则选取最左边的，并删除其中的一份（$XX \to X$）。问最后变成的串是多少。

算法分析

直接暴力似乎能碾过CF上的数据，但我们有更好的做法。

这题有两个较为显然的性质。首先，每次找到的子串的长度是不减的；其次，在删除的若干个长度相同的子串中，它们在删除前的串中的位置是不减的。证明比较显然，反证即可得出。

考虑如下问题：如何判断是否存在一个长度为L的X呢？一个经典的做法是，每L字符就设立一个观察点，那么长度为$2L$的XX一定会跨越两个相邻的观察点，我们对相邻两个观察点求最长公共前缀和最长公共后缀，如果加起来大于L那么就存在一个长度为L的合法的X。这一步的时间复杂度是$O(\frac{|S|}{L}\log L)$或者$O(\frac{|S|}{L})$（如果有预处理后缀数组和RMQ的话）。有解后的字符串重构的复杂度是$O(|S|)$的。

看起来复杂度仍然是$O(|S|^2)$的，没有任何改进。但我们把所有长度相同的删除的子串放在一起解决，而$1+2+...+2*\sqrt{|S|} > |S|$！所以不同的长度只有$O(\sqrt{|S|})$种，总的复杂度就是$O(|S|^{1.5}+|S|\log^2|S|)$。

63 Codeforces 319E Ping-Pong

题目简述

有若干个区间，能从区间(a,b)走到区间(c,d)的条件是$c < a < d$或者$c < b < d$，现在有$N(\leq 10^5)$个操作：插入一个区间，或者询问从一个区间出发能否走到另外一个区间，保证插入的区间的长度是递增的。

算法分析

我们把从某个区间走到另外一个区间的过程叫做连边。那么边的关系有两种：相交的话是双向的，而覆盖的话是单向的。而插入的区间的长度递增意味着插入的区间不会被任何一个已经插入的区间所覆盖，这样方便了我们后面的处理。考虑用并查集来维护所有双向边形成的联通分量，在一个分量里的所有区间都是可以相互到达的，我们就可以把它当成一个大区间来处理。

不妨用线段树来维护这些区间：线段树中某个结点存储覆盖该区间的联通分量的编号（可能有若干），而在插入区间(l,r)的时候，我们把l和r沿路上的所有联通分量与插入的区间合并，这样就得出了当前区间合并出来的大区间了，再将大区间插入线段树中。查询的时候判断是否在一个分量中或者两个分量可以通过单向边抵达即可。

时间复杂度$O(N \log N)$。

64 Codeforces 321D Ciel and Flipboard

题目简述

在$N*N$(N为奇数，$N \leq 33$)的整数矩阵上可以进行若干次操作，每次选取一个$x*x$的矩阵并把其中的数取反，其中$x = \frac{N+1}{2}$。问最后矩阵中元素的和最大是多少。

算法分析

设$f_{i,j}$表示在最后(i,j)格子上的数是否被翻转，1表示未翻转，-1表示翻转。

由于$x = \frac{N+1}{2}$，无论选取哪个矩阵翻转都会包含第x列和第x行。进一步观察后发现，对于某一行来说，在某一次操作中如果第x列被翻转了，那么第j列和第$j+x$列肯定有且只有一个被翻转，对于列也是如此。我们可以列出如下的两个等式：

$$f_{i,j} * f_{i,x} * f_{i,j+x} = 1$$
$$f_{i,j} * f_{x,j} * f_{i+x,j} = 1$$

我们看看如何利用这个性质。考虑枚举第x行的前x列的f状态，我们可以算出第x行的所有f值。接下来我们考虑前$x-1$行第x列的状态。在枚举第i行时显然我们可以顺带算出第$i+x$行的状态。我们可以看出，确定了这两个值之后，第i行和第$i+x$行的f取值就可以由这四个数贪心地算出了。由于这两行与其他行是没有相互关系的，我们可以贪心地取最值。最后就可以算出在当前枚举状态下能得到的最值了。

时间复杂度$O(2^x * N^2)$。

65 Codeforces 323B Tournament-Graph

题目简述

你需要构造一个$N(\leq 1000)$个节点的有向图，需要满足每对节点间有一条有向边相连，且无自环，并且任意两点的最短路不超过2；或者返回无解。

算法分析

考虑如下构造方法：对于节点i和$i+k$，如果k是偶数或者$k=1$则从i向$i+k$连边；否则从$i+k$向i连边。其正确性也比较显然。

唯一无解的情况是$N=4$。时间复杂度$O(N^2)$。

66 Codeforces 323C Two Permulations

题目简述

有两个长度为$N(\leq 10^6)$的排列和$M(\leq 2*10^5)$个询问，每次询问有多少个数在第一个排列中的位置在$[l1, r1]$之间，第二个排列中的位置在$[l2, r2]$之间，强制在线。

算法分析

我们可以把第一个排列调整成$1 \sim N$，这之后就是用主席树来解决的问题了。时间复杂度$O(N \log N)$。

67 Codeforces 325C Monsters and Diamonds

题目简述

有$N(\leq 10^5)$个物品和$M(\leq 10^5)$个规则，每种规则可以将一种物品分裂成至少一颗钻石和若干个物品。问对于每种物品来说，一开始拥有一个该物品时，最少（可能无解）和最多（可能是无穷大）能把该物品分成多少颗钻石。

算法分析

考虑求最小值。我们按照规则的拓补序来做最短路就可以求出第一问，不在拓补序中的点的答案就是-1。

第二问在该图中对点做一次DP就可以求出，如果途中出现了环则环上的点可以取到无穷大。

时间复杂度$O(M \log N)$。

68 Codeforces 325D Reclamation

题目简述

有一个$r * c(r, c \leq 3000)$的地图，把左边界和右边界粘起来使得形成一个圆柱，现在要不断地挖去其中的格子，要求任何时候都存在一条从最上方到最下方的路径(四联通)，如果某次操作不满足要求则不做，问最后有多少次操作是成功的。操作数$\leq 3 * 10^5$。

算法分析

直接维护空格的联通性比较困难，我们不妨转换思路，维护被挖掉的格子的联通性。由于空格是四联通的，被挖掉的格子的联通性是八联通的。把圆柱复制一遍拆成平面图形，那么判断上下边界是否联通的充要条件就是把某个格子挖空后它的两个对应位置是否联通，这个可以用并查集维护。如果对应位置联通的话我们就要把影响去掉。时间复杂度是$O(N * \alpha(N))$，

69 Codeforces 325E The Red Button

题目简述

一个$N(\leq 10^5)$个点的图，第$i(0 \leq i < N)$个点会向第$2 * i \bmod N$和第$(2 * i + 1) \bmod N$个点连单向边。求一条从0号点出发的哈密顿回路或者判断无解。

算法分析

首先考虑无解的情况。当N是奇数时，该图是无解的，因为0号点的前驱和$N-1$号点的前驱都只可能是$(N-1)/2$，显然是不能构成哈密顿回路的。

当N为偶数时，我们可以用如下的方法构造解。考虑对于点x和点$(x + N/2) \bmod N$来说，它们的后继都是相同的，显然选了一个的话另一个的取值就可以唯一确定了。我们可以用反正法证明这样做只会构成一个环。

时间复杂度$O(N)$。

70 Codeforces 329D The Evil Temple

题目简述

在一个$2N*2N$的棋盘中，每个格子可以是石头(”$<$”,”$>$”,””,”v”)或者空地(”.”)。一开始只有一个石头被激活，只有被激活的石头才会朝着它指向的方向前进。当它前进的下一个格子是另一个石头时，该石头会停在原地。如果是边界的话，将结束整个过程。如果某个石头移动了，那么在它停止移动时会发生碰撞。构造一个棋盘，使得出现$N^3 - N^2$次碰撞。

算法分析

需要这么多次碰撞我们显然需要构造循环。我们交错地给每行指定方向（蛇形）就可以达到尽量延长循环长度的目的。注意奇偶产生的影响。

71 Codeforces 329E Evil

题目简述

二维平面上有$N(1 \leq N \leq 10^5)$个点，两点之间的边权为它们的曼哈顿距离，求其哈密顿回路的长度最大是多少。

算法分析

我们找出x座标和y座标上的中位数，根据这两个值我们可以把整个平面割成四部分。分析最后的表达式我们不难得出，可能取到的最大值就是用排序后的序列的后一半的值减去前一半的值。

那么在什么情况下可以取到这个值呢？容易发现，如果所有点都分布在两个部分里，也就是对角线的两个部分的话，我们是一定可以取到最优值的。经过简单的代换我们可以得到，在对角的两个部分中，点的数量最多只会相差1。这样我们可以在两个部分中交替取点，就可以做到每次都是后一半的值减去前一半的值。

如果不是上面所说的情况呢？这样我们至少有一次做不到跨越对角地去取点了。在N为奇数的情况下，有些点是特殊的：它们恰好落站在中位数上。确切地说，如果两个座标上各有至少一个点在中位数上，我们还是可以取到上面所说的最值的。在不能跨越对角地取点时，我们先取那个在中位数上的点，再去到另外一个部分中，容易看出此时我们取到的仍然是最值。

当两维的中位数都只有一个点取到时（恰好落在分割线的交点上），最值是取不到的，因为有两次是不能做到跨越取点的，而我们只有一个可以承接在各部分之间的点，这样我们就需要做一些调整，具体来说就是在每维寻找与中位数最相近的值，然后用那个点来做承接点，这样我们的损失就是中位数与该值的绝对值差。

这样N为奇数的情况就讨论完了，当N为偶数时与上面说到的情况类似：我们寻找离中位数最近的值来做第二个承接点，也就是第$\frac{N}{2}+1$小的值所代表的点，这样它与中位数(第$\frac{N}{2}$小的)的差就是我们的损失，在两维中选一个较小的来做承接点就行了。

算法的每个部分都是$O(N)$的。

72 Codeforces 331C The Great Julya Calendar

题目简述

给定十进制正整数$N(\leq 10^{18})$，每次可以将它减去其十进制表示中的某位，问最少多少次操作可以将它变成0。

算法分析

一个显然的结论是，每次减去最大的一个数会是最优的，可以用数学归纳法证明答案的函数是不减的。

当N比较小时我们可以直接$O(N)$模拟方案。一个简单的优化是，考虑N的前一半位数和后一半位数，我们预处理在前一半的数在经过多少步后会减少，这需要我们预处理后一半的数在多少步后会变成0。显然在该过程中前一半的数不变，我们除了可以用后一半的数来减外，还可以用前一半里的最大的数去减，这样的状态是$O(\sqrt{N})$的，时间复杂度$O(\sqrt{N})$。

但即使如此，在$N = 10^{18}$时复杂度仍是不能接受的。实际上在上面的优化中，每次新变成的数的形式都是$a * 10^p - b$，其中p是前面数位的最后一位，所以其实这样逐位转移的状态数是$O(10 * 10 * digit)$的。至于状态的记录，我们可以直接用map来完成。

时间复杂度就是$O(10 * 10 * \log N)$。

73 Codeforces 331D Escaping on Beaveractor

题目简述

$B*B$二维网格上有$N(1 \leq B, N \leq 10^5)$个平行于座标轴且不相交的箭头，这些箭头标注了在其所占格子行走的方向。现在有M个询问，已知起点和初始方向，走$M(0 \leq M \leq 10^{15})$步后到达的位置，如果走出网格的话则输出最后在网格内的位置。

算法分析

不难发现在某个时刻后我们就在不断地兜圈。

我们算出从某个箭头的尾端出发会到达哪个箭头，或者是会走出网格。这个可以在排序后扫一遍，用线段树维护覆盖情况。同理我们可以顺便求出从某个询问点出发首先到达的是哪个箭头。

勾出图后剩下的事情也很明显了。在缩环后构成的树上我们做一遍DFS，维护树上的路径，也就是从某个箭头出发走到环的路径。求解询问时在路径上二分，或是走到环上后在环上二分。

剩下的事情就是努力码出来了。

74 Codeforces 331E Deja Vu

题目简述

在一个$N(N \leq 50)$个点M条边的有向图上，每条边上有一个有序的序列。如果一条路径上（不一定是简单路径）经过的点的序列和经过的边上的序列相同的话，那么这条路径就是合法的。统计对于每个在1到$2*N$之间的长度l，有多少条长度为l的合法路径。

算法分析

假设有一条长度为l的合法路径，上面有$l-1$条边。易得至少有一条边上的序列长度大于1。比较显然的另一点是，至少一条边满足，它的两个端点在它的序列上的位置是连续的。我们把这些边叫做“初始边”。

对于每条初始边，不难发现它往两边拓展的序列是确定的。一旦确定了序列的两端，我们就得到了一个合法的序列。容易证明在该序列中只有一条“初始边”。不妨把这个序列称为“主体”。

确定了“主体”之后，我们还可以拓展“主体”的两端。如果一条路径，它上面的边构成的序列与点构成的序列之间只差了序列的末端的话，我们就可以把它称作“头部”，因为它可以被接在主体的前面，并且序列仍然是完美的。相似地，只差了序列的首端的话，我们把它叫做“尾部”，因为它可以被接在主体的后面。我们把在主体前面加上若干个“头部”和在后面加上若干个“尾部”形成的序列称为“躯干”，这个可以用DP解决。

但是算出躯干后还不是最终的答案，因为两个躯干之间可以通过一条序列为空的边连接在一起。这个处理起来也很简单，做多一次DP就好了。

总的时间复杂度是$O(N^4)$。

75 Codeforces 332D Theft of Blueprints

题目简述

给出一个$N(\leq 2000)$个点的带权无向图，满足对于任意一个大小为$k(1 \leq k < N)$的顶点集合S，恰好有一个点与S每一个点都有边。令这个点为$v(S)$，并且对S进行操作的代价是S中每个点与$v(S)$的边权之和。现在求对于一个大小为k的子集操作代价的期望。

算法分析

当$k > 2$时，这个图必须要是完全图才能满足题目中的要求，证明可以通过反正法来完成。这样直接统计边的和，每条边选中的概率都是相同的，答案就是$2 * sum/N$。这个方法顺便解决了$k = 1$时的问题。

$k = 2$时也不是很困难：我们直接枚举$v(S)$，统计该点的度数和边权和，对于每条边，都有$deg_i - 1$个点与该边所对应的点的答案包含该边。答案就是$\frac{2*\sum_{i=1}^{N} sum_i*(deg_i-1)}{N*(N-1)}$。

时间复杂度$O(N^2)$。

76 Codeforces 332E Binary Key

题目简述

有一个长度为$N(\leq 10^6)$的匣子串S和一个长度为$K(\leq 2000)$的只由01组成的钥匙串T，如果$T[i \bmod K] = 1$那么将S_i加入信息串S'中。给定S，K和信息串$S'(|S'| \leq 200)$，求字典序最小的T。

算法分析

首先我们要确定在钥匙串中有多少个1。之后我们就可以贪心了：从后往前确定每一位的钥匙串，并且尽量填上1，这样在串的前面就有尽量多的0。

时间复杂度$O(N)$。

77 Codeforces 333C Lucky Tickets

题目简述

对于一个8位的数字串，如果在一些数的左边或者右边加入一些运算符号（“+”，“-”，“*”）和括号使得最终等于$K(\leq 10^4)$的话，那么它就是一个K-lucky Ticket。找出$M(\leq 3*10^5)$张不同的K-lucky Ticket。

题目简述

显然我们必须搜索，而且要构造性地去搜索。

考虑直接枚举序列的前一半，后一半直接通过前一半枚举符号算出来的结果得到，这样复杂度和实现的难度都比较低。

78 Codeforces 335D Rectangle And Square

题目简述

平面上有$N(\leq 10^5)$个互不相交的矩形，顶点均为$[0,3000]$的整数。问是否存在某个子集，满足该子集内矩形的并为正方形并输出方案。

算法分析

由于座标范围比较小，我们可以考虑以此下手问题。考虑枚举正方形的左下角，逐个检查边长为给定值时是否合法即可。

现在问题就变成判定性问题了。我们需要检查边界是否平滑和内部是否有空洞。对于前者我们可以预先处理边界上的区间和即可判断，而后者则是判断子矩形内的和是否为边长的平方，这些都是可以在预处理后在$O(1)$的时间内回答的。

时间复杂度$O(3000^2 + 3000 * N)$，不过上界是比较难达到的。官方题解有与座标范围无关的$O(N \log N)$算法，不过实现比较复杂。

79 Codeforces 335E Counting Skyscrapers

题目简述

大街上有很多摩天大楼，每栋楼的楼层数是一个正整数，高度为i层的概率为2^{-i}，每层的编号从0到$i-1$。大楼之间有天桥。如果两栋楼之间没有不超过x层的大楼且这两栋楼都有第x层的话，这两层之间就有一座天桥连接。Alice会从左到右数有多少座大楼，而Bob会从最左边的大楼开始，选择高度最高且不超过H的天桥并走到相连接的大楼直到走到最右的大楼。如果他在第i走了天桥他就会给他的计数器加上2^i。问题是已知Alice的答案，求Bob给出的答案的期望，或者已知Bob的答案，求Alice给出的答案的期望。

算法分析

如果是给出Bob的答案求Alice的期望的话，答案就是Bob给出的答案，计算贡献就可以算出期望了。

如果是给出Alice的答案求Bob的期望的话，事情就没有这么简单了。

我们不妨逐层计算贡献。我们计算新加入的桥并且减去被新加入的桥所覆盖的旧桥。我们从左到右计算某两座高楼之间存在一座桥的概率。对于高度H来说，如果两座楼之间的距离是L，那么在它们之间的$L-1$座楼的高度都不超过$H-1$，概率就是$1/2^{2*H}*(1-1/2^H)^{L-1}$。

那么被这座新桥所覆盖的旧桥呢？由于我们是逐层计算，我们只需计算高度为$H-1$的桥即可。又因为我们已知在新桥之间的楼的高度都不超过$H-1$，这之间每座楼的高度恰为$H-1$的概率就是$1/(2^H-1)$。我们只需知道这之间有多少座楼恰为$H-1$就能知道期望覆盖多少座旧桥，所以这部分的期望是$1+(L-1)/(2^H-1)$。一共有$N-L$座可能的新桥，对答案的贡献就是$(N-L)*(1/2^{2*H})*(1-1/2^H)^{L-1}*(2^i-2^{i-1}*(1+(L-1)/(2^H-1)))$。

统计所有可能的距离和高度后就是答案了。时间复杂度是$O(H*N)$，加上矩阵乘法后就是$O(H\log N)$，不过在本题中不需要。

80 Codeforces 335F Buy One, Get One Free

题目简述

商店里你有$N(1 \leq N \leq 5 * 10^5)$想买的派，每当你用原价买一个派时，你就可以免费选择一个比这个派便宜的派。问最少花多少钱可以吃到所有的派。

算法分析

花最少的钱就是要最大化免费获得的派的价值。

我们可以把所有价格相同的派归在一起，用二元组(价格，数量)来表示一种派。把所有的派按照价格降序排序后，可以轻松地得出一个DP方程。设$f_{i,j}$表示在前i种派中，免费获得了最多j个派时，最多能省多少钱。转移也很简单，在这里就不写了。时间复杂度显然太高，不能接受。

不妨分析一下，当我们新加入一种派时，f数组产生的变化。有两个个比较显然的事实：在f_i中，$f_{i,j}$随j的递增不递减。因此我们把f_i做差后的数列g也是非负整数。贪心的想一下很容易可以想到g数组也是不递增的，这也比较显然。

显然维护了g数组也就是维护了f数组。我们考虑维护g数组。假设前$i-1$种派一共有M个，新的派有y个，价值x元，那么我们最多可以免费获得$min(M,(M+y)/2)$个派。由于新加入的派比已经加入的派都要便宜，不难得出新加入的值是不会影响前面$max(0,min(M,(M+y)/2)-y)$个数的取值的。对于可能发生修改的位置，如果在对应位置上换成新的派更划算的话（g_j较小），显然应该选择新的派，所以我们把所有小于该派价格的g_j修改成x。

那么如果对应的位置是g_j更大呢？这意味着在那些新加入的可能的位置里的差就可能不是简单的认为是新派的价格，为了省更多的派我们可能要腾出价格更高的派以达到我们的效果，差值就要重新计算。不难得知这段区间是连续的。我们找出这段区间后可以抽象地认为存在一个价格为g_j的派，而现在我们要付g_j元来换取多省下一个派的钱，省下的价格就是$2*x-g_j$。如果这个值小于0的话显然是不会更优的。

这样我们就算出了新的g数组。正确性可以由DP的正确性推出。维护g的有序性后就可以解决本题了，最后的答案就是原来花的钱减去g的

和。

时间复杂度$O(N \log N)$。

81 Codeforces 338D GCD Table

题目简述

给定$N, M(\leq 10^{12})$和$K(\leq 10^4)$，问是否存在$x, y(1 \leq x \leq N, 1 \leq y \leq M-K+1)$使得$(x, y+i) = a_i(0 \leq i < K)$。

算法分析

x的最小可能取值显然是$[a_0, a_1, a_2, ..., a_{K-1}]$。
确定了x后，我们就只剩一个变量了。我们可以列出方程组

$$\begin{aligned} y &\equiv 0 \pmod{a_0} \\ y &\equiv -1 \pmod{a_1} \\ &\cdots \\ y &\equiv 1-K \pmod{a_{K-1}} \end{aligned}$$

解出可能的y后再判断可行性即可。
时间复杂度$O(K \log N)$。

82 Codeforces 338E Optimize!

题目简述

有一个长度为$N(\leq 1.5 * 10^5)$的序列a，问有多少个长度为M的子序列满足：和长度为M的序列b任意匹配后，每一对的和都不小于H。

算法分析

把b排序后，每个a中元素的合法区间就是从1开始连续的一段。

我们再求出每一个长度为M的子序列中，有多少个要和$b_1, b_2, ..., b_M$之前的元素匹配，记为c_i。显然，如果存在一个$c_i > i$的话，这个子序列是一定无解的，否则一定有解。我们现在的任务就变成了用线段树维护$c_i - i$的最小值。

时间复杂度$O(N \log M)$。

83 Codeforces 341E Candies Game

题目简述

有$n(\leq 1000)$个正整数$A_1...A_n(\sum A_i \leq 10^6)$，每次可以选择两个数$A_i, A_j(A_i \leq A_j)$，将$A_j$减去$A_i$并将$A_i$翻倍。构造一个操作序列使得最后有且仅有两个数大于0。

算法分析

考虑三个数$A, B, C(A \leq B \leq C)$，我们一定可以通过这三个数之间的操作构造出新的三个数$A', B', B'(A' \leq B' \leq C')$且$A' < A$。

不妨设$B = k * A + r(0 \leq r < A)$，考虑将$k$用二进制表达出来，我们显然可以不断对$A$进行操作，第$i$次操作时的$A$值就是$A * 2^{i-1}$，当$k$中对应位置的位为1时我们就对$A$和$B$进行一次操作。如果为0的话我们可以靠$C$来帮我们“跳过”这一位。显然在这段过程完成前都是满足$A \leq B \leq C$的，而在过程完成后B的值就变成了r，三个数中最小的数变小了。我们可以重复这段迭代过程直到一个数变成0为止。

对每一个数进行一次这样的操作后就可以构造出需要的序列了。时间复杂度就是构造出的序列的长度。

84 Codeforces 343E Pumping Stations

题目简述

有一个$N(\leq 200)$个点$M(\leq 1000)$条边的无向带权图，求一个点的排列使得排列中相邻两点的最大流之和最大。

算法简述

考虑如下的一个算法：任取点集中的两点S和T并计算出它们之间的最小割，把与S在残量网络上联通的点放在一起，不联通的放在一起，再重复之前的算法。我们记录下分裂的过程和每次的值，这样可以得到一棵带权的树。可以证明的是，在这棵树上任意两点的最小割和原图上任意两点的最小割是相同的。这棵树也被称为Gomory-Hu树。

构造出Gomory-Hu树后直接$O(N^2)$贪心就可以求出具体的排列了。时间复杂度$O(N * maxflow(N, M))$，其中$maxflow(N, M)$表示做一次N个点M条边的图的最大流所需的时间。

85 Codeforces 346E Doodle Jump

题目简述

给定$a, n, p, h(1\leqslant a\leqslant 10^9, 1\leqslant n < p\leqslant 10^9, 0\leqslant h\leqslant 10^9)$，对于$a * i \bmod p$，$(i \in [1..n])$，询问排序之后相邻两项的差的最大值是否$\leq h$。有$t(1\leqslant t\leqslant 10^4)$组数据。

算法分析

举例说明这题的做法。不妨设$a = 5, p = 23$，我们将依次生成的平台分组，如果某个平台的高度加上a超过p则新开一组。

0 5 10 15 20
2 7 12 17 22
4 9 14 19
1 6 11 16 21
3 8 13 18

不难发现在15∼20中有出现的平台在前面的区间(0∼5, 5∼10, 10∼15)中都有对应的平台，所以前面的空隙可以不用考虑。而在20以后的高度也是不会出现更大的高度差的，因为在第一组出现缺一个平台的的地方$(19 + 5 = 24 \geq P)$，该位置与上一组对应位置产生的空隙是不会在20以后的高度出现的。所以我们也不需要考虑20以后的高度。

这样我们就只需考虑15∼20之间的高度差即可。不难想到这段区间与0∼5之间的情况基本相同，这个问题的a值a'为$a - p \bmod a$，新的p值p'为原来的a值，新的n值n'为$a * n \operatorname{div} p$（需要判断此时的第$n'$个平台在原问题中有无出现，如果没有出现则$n'$要减1）。唯一的不同在于出现了高度为5的平台，但这个平台仅在新问题分组的组数为1时可能对答案1有影响。当分组数大于1时，设第一组最大高度为x，那么第二组第一个数为$x + a' - p'$，$x \sim p'$在第二组对应的区间就是$x + a' - p' \sim a'$，并且每当该区间有新的平台插入时，$x \sim p'$这段区间也一定有对应的平台插入。所以当组数大于1时我们不用考虑是否存在高度为p'的平台。

所以我们成功的将a值的规模缩成$a - p \bmod a$。但这样迭代的步数是可能达到$O(a)$级别的（考虑$p = a + 1$），所以仍然不行。在新问题中，所有平台的高度都是小于p'，也就是小于a的。假设$0 \le x \le y < p'$，我们容易得到$y - x = (p' - x) - (p' - y)$，也就是说，我们可以将所有的高度$x$变成$p' - x$而最终答案不变。设$x = p' * i = (a - p \bmod a) * i = (a - (p \bmod a) * i) \bmod a$，由上可得当$a' = p \bmod a$时答案仍然不变。那么我们可以选择$a - p \bmod a$与$p \bmod a$中较小的一个作为新的$a'$，这样$a'$最多仅为$a$的一半，迭代步数就是$O(\log a)$的了。总的时间复杂度就是$O(t \cdot \log a)$。

86 Google Code Jam 2009 Final D Wi-Fi Towers

题目简述

平面上有$N(\leq 500)$个塔，每个塔有发射半径和收益（可正可负），如果选用某个塔，那么在其发射半径内的塔也需要选用。求最大收益。

算法分析

赤裸裸的最大权闭合子图。
时间复杂度$O(N^3)$。

87 Google Code Jam 2011 Final B Rains Over Atlantis

题目简述

有一块$R*C(\leq 20)$的土地，每个格子上有一个非负的高度$(\leq 10^{15})$，土地外的边界高度为0。每一天由于下雨的关系，格子的高度会被流水侵蚀。算出当前格子的水位（水平面）后，每个格子的海拔会降至最低的与该格子相邻的格子的水平面，但降幅最多不超过M，问在多少天后所有格子的海拔都会变成0。

算法分析

由于有M的限制，答案可能会非常大，逐步模拟是不可行的。

不妨考虑在一次迭代的操作中模拟出许多天的情况。如果在这些日子中，所有没被淹没的格子的海拔都以最高速度下降，所有被淹没的格子的水位都以相同速度下降，并且没有被淹没的格子露出水面的话，我们就可以一次性地模拟出这些天数中水位和高度的变化情况。

那么我们会进行多少次这样的操作呢？首先我们容易发现，如果在某个时刻中某个格子露出了水面（没有被淹没）的话，在之后的时刻中它永远不会被再次淹没。而在上面的迭代结束时，至少有一个格子升出了水面，所以上面所说的快速迭代最多只会进行$R*C$次。

除了知道快速迭代次数的上限外，我们还要知道相邻两次快速迭代之间最多会做多少次普通的迭代。对于每个没被淹没的格子，我们向它流向的格子连边，而对于每个湖，我们向边界外连一条边。定义一个点是固定的，当且仅当从某个点出发沿着上面所说的连边走到边界外的过程中，相邻两点的高度差至少是M。如果每个点都是“固定”的，那么显然我们可以进行一次“快速迭代”。否则每天只会发生以下事件中的一种：某个格子露出了水面，或者是某个点的路径变成确定的了。由此我们知道在快速迭代之间最多只会发生$R*C$次普通迭代，总的迭代次数就是$O((RC)^2)$级别的了。每次迭代的时间复杂度是$O(RC\log RC)$(用最短路计算水位)，总的时间复杂度就是$O((RC)^3\log RC)$。由于常数较小，这个方法还是能在时限内出解的。

88 Google Code Jam 2012 Final D Twirling towards Freedom

题目简述

二维平面中有$n(\leq 5000)$个点，从原点出发，每一次可以绕其中一个点顺时针旋转90度，求最多走M步后最远能离原点多远。

算法分析

考虑用复平面表示问题，那么顺时针旋转90度就是乘上$-\mathrm{i}$。设第i次绕a_i旋转后停在P_i，那么$P_i - a_i = -\mathrm{i} * (P_{i-1} - a_i)$。

我们化简式子之后，就可以推出

$$P_m = ((1+\mathrm{i})*(a_m+a_{m-4}+a_{m-8}+...)+(1-\mathrm{i})*(a_{m-1}+a_{m-5}+am-9+...)$$

$$+(-1-\mathrm{i})*(a_{m-2}+a_{m-6}+a_{m-10}+...)+(-1+\mathrm{i})*(a_{m-3}+a_{m-6}+a_{m-9}+...)$$

不妨设X表示从原点能到达的最远的点。更一般地，我们设X_v表示在从原点出发，在方向v上能取到的最远的点。显然X肯定是某个方向v上的X_v。

那么我们现在的任务就变成了找出某个方向v，使得P_m在方向v的长度最大。由于我们此时考虑的是投影的长度，是一维的东西，那么对于系数相同的点（a_m&a_{m-4}之类的），我们的选择显然是相同的。

现在我们需要做的就是对于一堆点a，在方向v上最远的点是什么。看起来直接扫就行了。但是由于我们有无限个不同的方向，直接扫是不可行的。既然对于某个具体的方向求解不能解决问题，那么我们可以考虑对于某个点，在哪些方向上的距离是这个点集中最大的。

这个问题我们可以用凸包来解决。显然如果某个点不在该点集的凸包上，它是一定不会成为最优解的。而凸包上相邻两点的向量差就是最优解的分解线。

现在我们求出了一个点集时的最优值，但我们有四个不同的点集（常系数不同），我们要把最优解组合起来。这里有$O(n^2)$的做法，也有$O(n\log n)$的做法。我们可以把所有的分解线按照极角排序，从小到大扫

一遍，对于某个事件点，我们修改对应点集的最优值并计算当前的答案并更新最优解。这样的时间复杂度就是$O(n\log n)$的了。

89 USACO JAN07 Cow School

题目简述

有$N(\leq 50000)$个物品，每个物品有两个属性$T_i, P_i(0 \leq Ti \leq P_i < 40000, 0 < P_i)$，现在按照$T_i/P_i$的升序给物品排序，问对于哪些$D$来说，去掉前$D$个数后的$\frac{\Sigma T_i}{\Sigma P_i}$并不是所有在去掉$D$个数的情况下所能取到的最值。

算法分析

假设我们确定了D，并且去掉前D个数后的比值是r，那么我们判断的依据就是是否存在一对j, k使得$T_j - r * P_j > T_k - r * P_k(j \leq D, k > D)$。

我们可以将上式理解成点(T_j, P_j)被斜率为r的直线所截得到的在y轴上的截距。这启示我们不妨考虑从几何的角度来考虑问题。

分析以下当D增加时会发生什么事情。显然随着D的增加，比值r是会递增的。因此，对于所有位于新插入点上方的点来说，它们以后都是不能成为最优解的了。这样凸壳的维护的时间就是$O(1)$的了。在凸壳上找出截距最大的点就是简单的事情了。相似地，我们也可以用类似的方法找出截距最小的点。

时间复杂度$O(N \log N)$。

90 USACO DEC07 Best Cow Line

题目简述

有一个字符串$S(|S| \leq 30000)$，每次可以从串首或串尾取出一个字符加入新串串尾中，问可能得到的字典序最小的串。

算法分析

一个显然的思路是贪心：直接判断当前串首和串尾字符的大小，选出较小的加入新串。

但如果二者相同呢？我们显然不能随便选一个，而是应该继续比较第二个字符和倒数第二个字符谁小，如果是第二个比较小则取出串首，如果倒数第二个较小则取出串尾，相同则继续比较，依此类推。

这样我们实际上做的事情是比较一个后缀和一个前缀的倒序的字典序大小，将串S的反转加在S的后面，用后缀数组排序后就出解了。

时间复杂度$O(|S|)$。

91 USACO MAR08 Land Acquisition

题目简述

有$N(\leq 5*10^4)$个二元组(x_i,y_i)，现在要把二元组分成若干组，每组的花费是$max(x_i)*max(y_i)$，问最小总花费。

算法分析

显然，如果存在$x_i \leq x_j$且$y_i \leq y_j$的i话，那么这个二元组是不会对答案造成影响的，可以删去。对剩下的二元组按照x升序排序，那么y值就是降序了。

不难说明排序后，最优解中被分在一组的二元组肯定是连续的一段。那么我们就可以DP了。

这个DP可以用经典的斜率优化做到$O(N)$，总的时间复杂度就是$O(N\log N)$。

92 USACO JAN09 Safe Travel

题目简述

有一个$N(\leq 10^5)$个点$M(\leq 2*10^5)$条带权无向边的图，保证从1号点出发到各个点的最短路存在且唯一。现在要求从1号点到各个点的不经过从1号点到该点的最短路中的最后一条边的情况下的最短路是多少。

算法分析

首先我们建出最短路树，由题目可知该树的形态是唯一的。题目中要求的就是不经过某点和该点父亲之间的边的情况下的最短路。

其次我们需要证明的是，能够成为最优答案的路径上只有一条不在最短路树上的边，简单的替换一下就发现存在一个不会更劣的选择。

有了这个之后题目就好解决了。我们可以给每个点维护一个堆，堆里的每个元素代表选择某条边作为那条不在最短路树上的边时的答案，堆中的最小值即是答案。该边的两个点在最短路上的状态一定是一个点在所求的点的子树内，一个点在所求的点的子树外，所以该堆可以由其儿子的堆合并而来，在合并堆的时候打上标记（由儿子走到父亲的边权）即可。至于合法性，我们直接判断两点是否都在该点子树内即可。当某条边变的不合法时在其祖先的状态中也是不合法的，我们可以把它懒惰删除掉。

时间复杂度$O(M \log N)$。

93 USACO MAR09 Cleaning Up

题目简述

有一个长度为$N(\leq 40000)$的序列，每个元素都是$[1, M(\leq 40000)]$之间的整数。现在要将序列分成若干段，每段的代价是该段内有多少种不同的元素的平方。求最小总代价。

算法分析

总代价的上界显然是N，因为每个元素单独成为一段的代价就是1，总代价为N。也就是说，如果有一段有超过$\sqrt{N}$个不同的元素，那么显然不可能成为最优解。

设f_i表示做完前i个数后的最小总代价，显然$f_{i-1} \leq f_i$。转移方程也很显然：$f_i = min(f_j + w_{j+1,i}{}^2)$，其中$w_{l,r}$表示$l$到$r$中有几个不同的数。

这样看这个方程似乎不好优化，但由于w的取值最多只会到$\sqrt{N}$，我们可以维护$w_{l,i} = k$时l的最小值，这样状态转移就是$O(\sqrt{N})$的了。

时间复杂度$O(N^{1.5})$。

94 USACO OPEN09 Tower of Hay

题目简述

一个有$N(\leq 10^6)$个正整数的序列，求最多能够分成多少段，使得每段内数的和不大于上一段的和。

算法分析

考虑倒序贪心考虑问题。如果要分成尽量多的块的话，我们显然应该尽量使每段内的和最小，这个证明也是比较简单的。

这样我们就可以DP了。而这个DP是决策单调的，决策的插入也是单调的，我们可以用单调队列优化。

时间复杂度$O(N)$。

95 USACO OPEN10 Triangle Counting

题目简述

给出$N(\leq 10^5)$个笛卡尔坐标系上的整点，统计有多少三角形包含原点$(0,0)$。

算法分析

不妨考虑统计有多少个三角形不包含原点。

我们按照极角排序。如果三点构成的三角形不包含原点的话，那么它们一定在其中一点与原点连线的同一侧，并且这样的点有且仅有一个。

只需要单调地扫描，维护同侧的点有多少个就可以出解了。

时间复杂度$O(N \log N)$。

96 USACO DEC10 Threatening Letter

题目简述

给定两个串，长度不超过50000，每次可以复制第一个串中的一个子串，问最少需要多少次复制可以构造出第二个串。

算法分析

一个显然的贪心策略是：从某个位置开始，每次我们都选极大的合法子串，这样的选择一定是最优的。

那么用后缀自动机判断可行性就可以了。

时间复杂度是线性的。

97 USACO MAR12 Cows in a Skyscraper

题目简述

有$N(\leq 18)$个数，问最少可以把它们分成多少组，使得每组的和都不超过W。

算法分析

直接找出每个极大子集后BFS出答案是可以通过数据的，但这样的复杂度是$O(3^N)$，不够出色。

设$f_{S,i}$表示我们将子集S分成了i组，第i组的和最小是多少（如果可行的话）。转移也很显然，枚举新加入第i组的是哪个数即可。

时间复杂度$O(N^2 * 2^N)$。

98 USACO DEC12 First!

题目简述

有$N(\leq 30000)$个仅有小写字母的字符串（总字符数不超过$3*10^5$），现在问哪些串可以通过改变字母表的顺序是该串变成字典序最小的。

算法分析

对所有串建出Trie树。我们可以给字母之间建立一个有向图，字母a向字母b连边表示a一定要在b的前面。对于每个串我们都可以建出一个这样的图。如果该图是DAG的话那么就是有解，否则肯定无解。

时间复杂度$O(\sum |S| + N * |\omega|^2)$。

99 USACO OPEN13 Figure Eight

题目简述

在$N * M(N, M \leq 300)$的网格上，每个格子不是空地就是障碍。现在要求求出分数最高的数字8。数字8的定义如下：

数字8由上下两个矩形构成。

数字8的上下两个矩形都满足至少有一个单元格在矩形内部。

数字8顶部的矩形的底边必须为底部矩形顶边的子集。

数字8只能出现在没有障碍的位置上。

数字8的得分为上矩形和下矩形的面积的乘积。

算法分析

考虑确定下矩形的顶边，因为这样同时也确定了上矩形底边的范围，并且将上下矩形分开考虑。

确定了矩形的底边（或顶边），我们可以通过预处理快速地推出它的另外一边最远放置的位置。以此我们可以直接求出下矩形的位置。用类似的方法确定上矩形的位置后，用DP统计被该顶边包含的底边的最大可能面积。

时间复杂度$O(NM^2)$。

100 USACO OPEN13 Photo

题目简述

有一个长度为$N(\leq 10^5)$的01串，现在有$M(\leq 10^5)$个约束，每个约束表示在$[l,r]$内有且仅有一个1，问该串最多有多少个1。

算法分析

我们可以设dp_i表示第i个位置为1时，前i个位置最多有多少个1。显然能够转移到i的是一段连续的区间，而且区间的两端都是单调的。我们用单调队列维护该区间内的最大值即可以将转移优化至线性。

时间复杂度$O(N)$。