IOI2015国家集训队第一次作业 试题泛做

湖南省长沙市长郡中学 陈胤伯

Contents

1 Rhomebus

1.1 试题来源

Codeforces 263E

1.2 题目大意

给一个n行m列$(n, m \leq 1000)$的数字矩阵，再给定一个正数k。

请你找到一组(x, y)，满足:

- $k \leq x \leq n - k + 1$
- $k \leq y \leq m - k + 1$

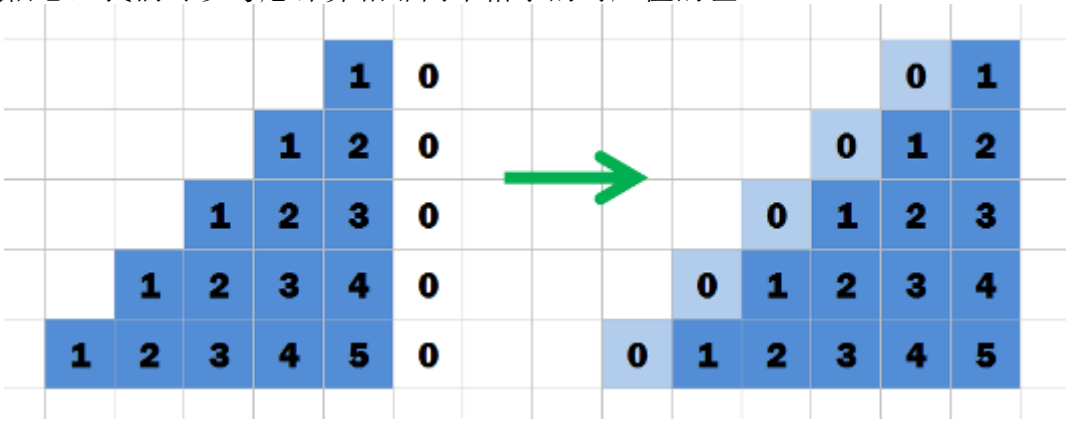
最大化：

$$\sum_{i=1}^{n}\sum_{j=1}^{m} a_{i,j} * \max(0, k - |i - x| - |j - y|)$$

1.3 算法分析

这道题实际上等价于每个格子向外扩展一个宽为k的加权菱形，然后对应位置相乘后相加，要找到和最大的那个格子。

一个直接的想法是枚举每个格子，然后计算这个和。为了提高效率，合并重复信息，我们可以考虑计算相邻两个格子的对应值的差。

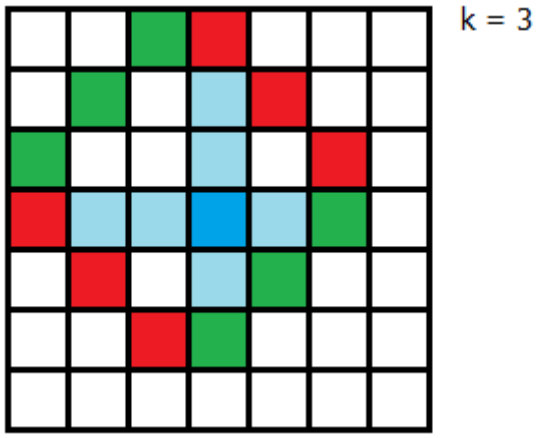


如图所示，我们只考虑一个三角形部分（另外三部分同理）。移动一格之后，少了一个三角形的区域和，多了一个加权的竖条和，而这两部分可以通过递推预处理出来。

一个加权竖条和只比它上面的加权竖条和多了一条竖着的区间和，可以竖着递推每个格子的加权竖条和。

一个三角形区域和（图中的情况）只比它左边的三角形区域和少了一条斜向区间和、多了一条竖向区间和，可以横着递推每个格子的三角形区域和。

事实上这道题还有一个更为简单的做法，记$S_{i,j}$为格子(i, j)的前缀和，我们尝试用S拼凑出所求的加权菱形。



如图所示，绿色的表示加上的S，红色的表示减去的S，用这样四个斜条区间和就成功拼凑出了加权菱形。

1.4 时空复杂度

时间复杂度：$O(nm)$

空间复杂度：$O(nm)$

2 Distinct Paths

2.1 试题来源

Codeforces 293B

2.2 题目大意

给一个n行m列$(n,m\le 1000)$的棋盘，你有k $(k\le 10)$种颜色。

棋盘上的一些格子已经染好了$1\sim k$的颜色，你要给剩下未染色的格子们染上$1\sim k$的颜色，满足：任意一条从棋盘左上角到右下角的路径不经过颜色相同的两个格子（每一步只能向右或向下走一格）。求合法的染色方案数。

2.3 算法分析

稍作分析可以发现，题目的限制条件等价于：任意两个同色格子不互为左上、右下的位置关系。

我们可以感受到，这是一个很强的限制，由于k不大，看上去n、m太大应该是无解的。我们随便找一条从左上到右下的路径，显然长度为$n+m-1$，那么若$k<n+m-1$，光是这条路径上颜色就不够用了。所以若方案数不为0，则$n+m\le k+1$。这样一来棋盘大小就很小了，搜索算法成为一个不错的选择。

在暴力枚举每一个格子颜色的基础上，还需要加上一些剪枝。

首先，每个格子存下其对应的前缀里所有数字的出现情况（二进制压位）。

然后，用最小表示法的思想，我们强制规定：标号越小的颜色第一次出现越早。对于棋盘上原本存在的颜色，我们记录下对应的替换关系，这样搜索出来的每一个方案，都对应了一类合法方案。设一共填了$used$种颜色，有a种对应的是棋盘上原本存在的颜色，那么剩下$used-a$种颜色是可以从$k-a$种颜色里任取的，也就是有$\prod_{i=1}^{used-a}(k-a-i+1)$种方案。

优化之后的搜索经测试是可以通过全部数据的。

2.4 时空复杂度

时间复杂度：未知（搜索算法）

空间复杂度：$O(nm)$

3 Number Challenge

3.1 试题来源

Codeforces 235E

3.2 题目大意

令$d(x)$表示x的正约数个数，计算：

$$\sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{c} d(i*j*k)$$

$$1 \le a,b,c \le 2000$$

3.3 算法分析

定理 3.1.

$$\sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{c} d(i*j*k) = \sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{c} [(i,j)=1 \text{ and } (j,k)=1 \text{ and } (k,i)=1] * \lfloor\frac{a}{i}\rfloor * \lfloor\frac{b}{j}\rfloor * \lfloor\frac{c}{k}\rfloor$$

证明 3.1.

$$f(a,b,c) = \sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{c} d(i*j*k)$$

$$g(a,b,c) = \sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{c} [(i,j)=1 \text{ and } (j,k)=1 \text{ and } (k,i)=1] * \lfloor\frac{a}{i}\rfloor * \lfloor\frac{b}{j}\rfloor * \lfloor\frac{c}{k}\rfloor$$

那么，只需要证明：

$$f(a,b,c) - f(a-1,b,c)... + f(a,b-1,c-1) - f(a-1,b-1,c-1)$$

$$= g(a,b,c) - g(a-1,b,c)... + g(a,b-1,c-1) - g(a-1,b-1,c-1)$$

左边$= d(i * j * k)$

右边

$$= \sum_{(i,j)=1 \text{ and } (j,k)=1 \text{ and } (k,i)=1} (\lfloor \frac{a}{i} \rfloor - \lfloor \frac{a-1}{i} \rfloor) * (\lfloor \frac{b}{i} \rfloor - \lfloor \frac{b-1}{i} \rfloor) * (\lfloor \frac{c}{i} \rfloor - \lfloor \frac{c-1}{i} \rfloor)$$

=互质三元组(i, j, k)且满足$a \bmod i = 0$ and $b \bmod j = 0$ and $c \bmod k = 0$ 的个数。

考虑每一个质数p，设x是满足$p^x \mid a$的最大的x，y、z分别对b、c进行类似定义，那么：

对左边，贡献了一个因子$x + y + z + 1$。

对右边，同样也是贡献了一个因子$x + y + z + 1$。

因此左边=右边。

接下来我们用定理所给的式子来计算答案。

假如枚举了i，我们需要快速统计：

$$\sum_{j=1}^{b} \sum_{k=1}^{c} [(j,k) = 1] * \lfloor \frac{b}{j} \rfloor * \lfloor \frac{c}{k} \rfloor \tag{1}$$

$$= \sum_{j=1}^{b} \sum_{k=1}^{c} \lfloor \frac{b}{j} \rfloor * \lfloor \frac{c}{k} \rfloor * \sum_{d \mid (j,k)} \mu(d) \tag{2}$$

$$= \sum_{d} \mu(d) * F(\lfloor \frac{b}{d} \rfloor) * F(\lfloor \frac{c}{d} \rfloor) \tag{3}$$

其中：

$$F(n) = \sum_{i=1}^{n} \lfloor \frac{n}{i} \rfloor$$

容易发现，预处理$\mu(d)$后计算这部分是$O(\frac{n}{1} + \frac{n}{2} + ...) = O(n \log n)$ 的。

当然，目前的计算式还漏了一个条件：$(j, i) = 1$ and $(k, i) = 1$。好在我们是暴力统计，只要判断外层枚举的d 和内层枚举的除数分别与i是否互质即可。为了不影响复杂度，我们还需要用$O(n^2 \log n)$ 的时间预处理$0 \sim 2000$所有数字两两的gcd。

3.4 时空复杂度

时间复杂度：$O(n^2 \log n)$

空间复杂度：$O(n^2)$

4 The Red Button

4.1 试题来源

Codeforces 325E

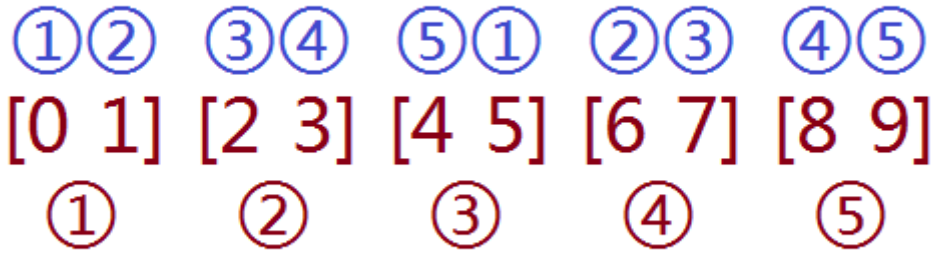
4.2 题目大意

有n ($n \leq 10^6$)个点，标号为$0 \sim n-1$。点i可以走到点$2*i \bmod n$和点$2*i+1 \bmod n$，请你找一条哈密尔顿回路。

4.3 算法分析

观察一下小数据，我们可以发现奇数都是无解的。稍加思考便可以发现原因：首先，每个点在回路中有且仅有一个前驱、一个后继。由于$i \leq n-1$，故$2*i+1 \leq 2*n-1$，所以0 的前驱只能是$\frac{n}{2}$或$\frac{n-1}{2}$，这取决于n的奇偶性。我们再考虑$n-1$ 的前驱，只有$\frac{n-1}{2}$或$\frac{n-2}{2}$（不可能是它自己，不然它就从回路中孤立出来了），同样也取决于n的奇偶性。n一旦是奇数，这意味着0和$n-1$的前驱都只能是$\frac{n-1}{2}$，而每个点只能有一个后继，所以是不存在合法解的。

现在只考虑n为偶数的情况，为了进一步思考，我们写下每个点的出边并进行观察。



如图所示，如果把$0 \sim n-1$两两分成一组，再给每一组一个标号，那么每个点的出边都连向同一组中的两个点。我们把每个点的出边指向的组写成一排，发现是$1 \sim n$重复两次。

考虑到要构造哈密尔顿回路，我们不妨从欧拉回路入手。现在把每个组看

成一个大点，组与组之间连边，那么每个大点的入度=出度= 2，并且大点之间互相连通，也就是一定存在欧拉回路。

我们dfs求出一个边的出栈序列，就得到了大点们的欧拉回路。有了这个欧拉回路，把每条边对应到相应的出发点上去，就得到了题目所求的哈密尔顿回路。

4.4 时空复杂度

时间复杂度：$O(n)$

空间复杂度：$O(n)$

5 Dividing Kingdom

5.1 试题来源

Codeforces 260E

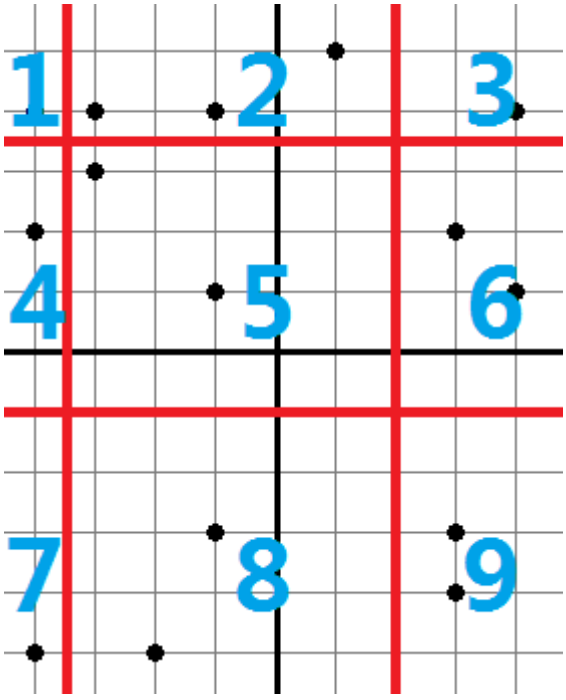
5.2 题目大意

平面直角坐标系内给定n ($n \leq 10^5$)个点，点i坐标为(x_i, y_i) ($-10^9 \leq x_i, y_i \leq 10^9$)，再给定9个数字$a_{1\sim9}$。

你需要画两条平行于x轴的直线、两条平行于y轴的直线（线与线不能重合，且每条线不能穿过给定的点），把平面分成9个区域，使得这9个区域内分别拥有的点的个数与$a_{1\sim9}$相等。（匹配的时候可以调换顺序）

5.3 算法分析

由于匹配时可以调换顺序这点十分难处理，我们不妨先$O(9!)$枚举匹配的顺序。



如图所示，我们给这9个区域编号，每个区域内部的点数是已知的。这样一

来，通过区域1、4、7的点数和，我们可以确定靠左的竖线。用类似的方式，可以最终确定出四条线。

接下来开始验证是否可行。方法比较多，但离不开一件事情：查询一个矩形内点的个数。这是一个经典问题，可以用函数式线段树来统计。笔者的方法是：查询{7}、{7,8}、{7,4}、{7,4,8,5}这四个前缀矩形，根据这四个值可以推出其他所有区间，

5.4 时空复杂度

时间复杂度：$O(9! \times \log n)$

空间复杂度：$O(n \log n)$

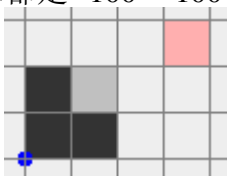
6 Princess and Her Shadow

6.1 试题来源

Codeforces 317E

6.2 题目大意

一个平面直角坐标系内有你和影子以及m ($m \le 400$)个障碍物，所有物体的坐标都是$-100 \sim 100$ 的整点，且互不重合。



每一步你可以朝上下左右中的一个方向走一格，然后影子会做相同的动作。不过，影子若下一步会走上障碍则不会移动，而你不可以走上障碍。

请你找到一组合法方案，使得最终你和你的影子在同一格。

（要求步数不超过10^6）

6.3 算法分析

最终的目的是抓到影子，那么可以考虑一直沿最短路朝影子走去，直到抓住影子为止。下面我们来分析这样做的可行性。

如果一开始你和影子不连通，那么显然无解。

否则，假设当前你到影子的最短路长度为l，那么之后你到影子的最短路肯定不会超过l，因为至少你可以沿着影子的上一回合的路径走。如果最短路长度一直减少，那么抓到影子就指日可待了。

问题在于，最短路长度不减该怎么办。

若最短路长度不减，则你的移动向量和影子的移动向量是相同的，重复多次后，要么影子因为碰壁导致你的最短路长度下降，要么你们一起走到了很远的空旷地方。前者并不值得我们担心，关键是后者。考虑你和影子处在很远的地方，如何抓住影子。

如果整张地图没有任何障碍物了，那肯定是抓不住影子的。

否则，我们可以利用地图边界上的障碍物，也就是x、y坐标最大(最小)的那些，来把自己和影子蹭到一起。因为你可以带着影子在空旷地带自由走动而保持相对位置不变，只要根据需要，走到对应的边界上，然后蹭齐，再回到远处，于是你也能成功抓到影子。

代码实现时需要十分细致，注意讨论多种情况。

6.4 时空复杂度

时间复杂度：未知（构造算法）

空间复杂度：$O(x_{max} \times y_{max} + m)$

7 Reclamation

7.1 试题来源

Codeforces 325D

7.2 题目大意

有一片表示成$r*c$ $(r,c \le 3000)$个格子的环形海洋，可以想象成一个圆柱的侧面。

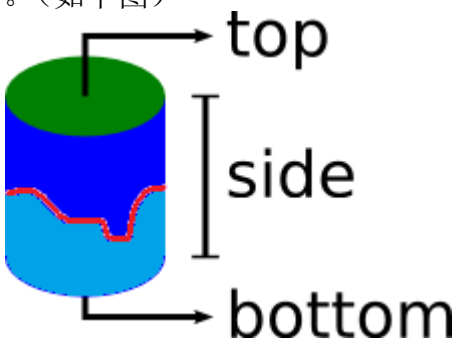
你接到了n $(n \le 3*10^5)$个请求（按时间顺序给出），第i个请求要把(x_i,y_i)处的格子填成土地。

你需要保证任意时刻海上航路畅通，即存在至少一条海洋里的通路从底部通往顶部，因此你需要对于每个请求作出是否接受的判断，接受的请求将被执行。

输出最后接受的请求总数。

7.3 算法分析

我们想象一个立体的环形网格图，考虑一下，如果底部不能走到顶部，我们从底部做FloodFill，一定能得到一个土地组成的环把底部到顶部的路径给隔断了。（如下图）



那么现在就是要在这个立体的环形网格图里找环。直接用并查集就行了？我们发现只考虑环是不行的，这个阻隔环还必须转角非零，也就是至少要转过一圈。

我们把环形网格断开成一个$r*c$的网格，然后复制一份连在右边，填土地时两边都填上。当且仅当存在点(x,y)可以到达其复制点$(x,m+y)$时，阻隔环出现。

虽然我们不能随时关注到每个点是否能到达其复制点，但注意到，加入新的土地时，倘若形成了环，那么新的土地所在的点必然在形成的环上。于是，我们只要判断加入新的两个点后，它们是否属于同一连通块，如果是就拒绝该请求并撤销操作，否则接受改请求并给答案加一。

7.4 时空复杂度

时间复杂度：$O(n * A(n))$

空间复杂度：$O(r * c)$

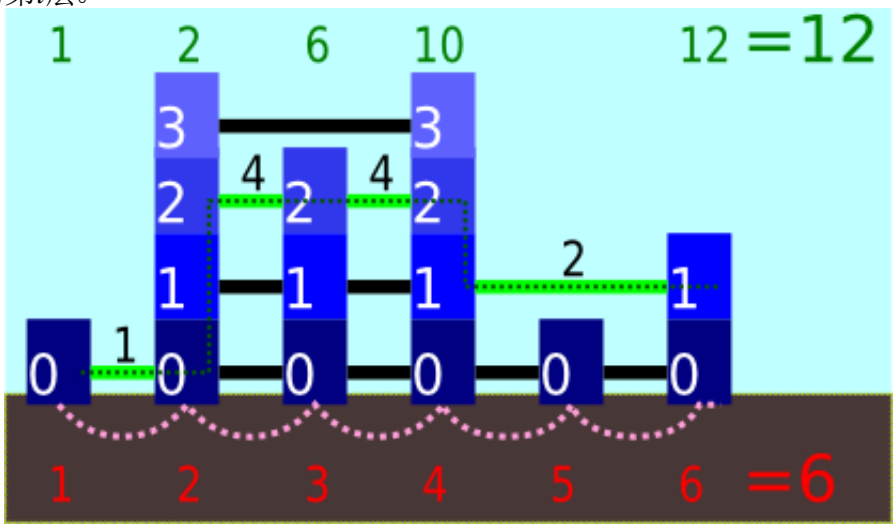
8 Counting Skyscrapers

8.1 试题来源

Codeforces 335E

8.2 题目大意

有一排摩天大楼，数量在2到314!中均匀随机选择。对于每一栋摩天大楼，它总层数为i的概率是$\frac{1}{2^{i+1}}$，层数从0开始标号。假如两栋摩天大楼的总层数都不小于h，且它们之间所有摩天大楼的总层数都小于h，那么有一条滑索连接着它们的第i层。



定义计数器A的值等于摩天大楼的数目。

定义计数器B，初始值为1，然后一个人拿着B从第一栋大楼出发，每次沿着最高的滑索前往下一栋大楼，并给B计数器加上$2^l v$，其中lv为滑索的高度。顺便一提，这个人有恐高症，因此他会忽略所有高度大于h的滑索和楼层。

现在给定A、B中一个的值n ($n \leq 30000$)，以及参数h ($h \leq 30$)，请你求另一个计数器的期望。

8.3 算法分析

先考虑给B如何求A。通过对样例的观察，我们发现$E(A) = B$，考虑如何证明。

考虑那个拿着B的人从第t楼使用了一次滑索，如果我们可以证明，这次经过的楼房个数的期望恰好为2^t，那就完事了。我们设一栋楼房高度小于t的概率为p，显然$p = 1 - 2^{-t}$。考虑中间的楼房高度都要小于t，最后一栋楼房高度不小t，那么期望经过的楼房数为：$1 + p + p^2 + p^3 + ... = \frac{1}{1-p}$，带入$p$后就是$2^t$，恰好是我们计数器加的值。

下面考虑给A求B，即已知大楼总数，求B计数器的期望。接下来的讨论均默认大楼高度不超过h。

我们先考虑滑索的生成方式：想象一条水平扫描线从下扫到上，对于每个高度，连接所有相邻的大楼。观察拿着B的人的行动路线，发现他恰好走过了每一条“顶部”的滑索。接下来，我们考虑依次统计每一种高度的滑索对计数器期望的贡献。鉴于低的会被高的覆盖，这样给统计带来了诸多不便，不妨在每一层加上贡献的同时减去下面一层被覆盖掉的贡献。

对于某个高度h，设某条滑索的长度为L，首先它的位置有$n - L$种可能。滑索两端的大楼高度不小于h，概率为$\frac{1}{2^{h\times 2}}$，中间所有大楼的高度小于h，概率为$(1 - \frac{1}{2^h})^{L-1}$。对于加上的贡献，就是$2^i$。对于减去的贡献，即$2^{i-1}$乘上这条长为$L$的滑索下面高度为$h - 1$ 的大楼的期望数目。考虑一栋大楼，高度为$h - 1$的概率是$\frac{1}{2^h}$，由于之前我们是在“高度小于h”的前提下讨论的，所以算上条件的概率应该是$\frac{1}{2^h} \div (1 - \frac{1}{2^h}) = \frac{1}{2^h-1}$，这个概率乘上$(L - 1)$就是下面高度为$h - 1$的大楼的期望数目了。

最后的计算式为：

$$\sum_{L=1}^{n-1}\sum_{h=0}^{maxh}\frac{1}{2^{h\times 2}}\times(1-\frac{1}{2^h})^{L-1}\times(2^i-2^{i-1}\times\frac{L-1}{2^h-1})$$

注意$h = 0$时要特判处理。值得一提的是，这个计算式是可以用矩阵优化的。

8.4 时空复杂度

时间复杂度：$O(nh)$

空间复杂度：$O(1)$

9 Levko and Sets

9.1 试题来源

Codeforces 360D

9.2 题目大意

给两个数列$a_{1\sim n}$ ($n \le 10^4$)和$b_{1\sim m}$ ($m \le 10^5$)以及一个质数p ($p \le 10^9$)。

有n个集合，集合i的生成方式为：最初只有一个数 1，每次任意选取集合中的一个元素x和一个j，如果$x * a_i^{b_j}$不在集合中，就加入集合，一直这样做直到不能加入新的数字为止。

求这n个集合并的大小。

9.3 算法分析

由于p是质数，$x^{p-1} \bmod p = 1$。那么对于某一个集合，用b_j凑出来的指数是在模$p-1$意义下的，所以所有能凑出的指数均可表示为$z * tb$，其中z是任意非负整数，$tb = \gcd(b_1, b_2, b_3, ..., p-1)$。

设模p意义下的原根为g，设$a_i = g^{r_i}$，那么我们只用关心指数在模$p-1$意义下的取值，即$z * r_i * tb \bmod (p-1)$的取值，显然就是$[0, p-1)$中$\gcd(r_i * tb, p-1)$的倍数们。令f_i表示$[0, p-1)$有多少个数字的最大约数为i，递推方程为：

$$f_i = \frac{p-1}{i} - \sum_{i \mid d} f_d$$

至于求r_i，我们可以暴力枚举$p-1$的约数，找到$a_i^e \bmod (p-1) = 0$最小的e，然后$r_i = \frac{p-1}{e}$。

9.4 时空复杂度

时间复杂度：$O(n * d(p-1) + d(p-1)^2)$

空间复杂度：$O(n+m)$

10 Three Swaps

10.1 试题来源

Codeforces 339E

10.2 题目大意

有一个长为n ($n \le 1000$)的数列，第i个数为i，你可以进行至多 3 次区间翻转操作，要求变成给定的目标序列$a_{1\sim n}$。

10.3 算法分析

我们考虑目标序列怎么翻转回去。

首先，去掉已经对上位的前缀和后缀，显然任何翻转都不会涉及到这些部分，因为涉及到了不会对我们的归位有任何好处。

假设现在待归位的区间是$a_{l\sim r}$，因为l和r位置上都还没匹配，而第二步至多归位一个端点（若第二步归位两个端点，意味着翻转了$l \sim r$整个区间，而这样的操作提到第一步做是等价的），所以第一步和最后一步一定有一步是“把l翻转到左端”或者“把r翻转到右端”。

假如是第一步，我们就可以直接枚举第一步翻转了（只有两种情况）。如果是最后一步怎么办呢？我们可以最后把问题倒过来再做一次。

第二步进行类似的枚举，这样最终一共讨论的情况是常数级别的。

10.4 时空复杂度

时间复杂度：$O(n)$

空间复杂度：$O(n)$

11 Ladies' Shop

11.1 试题来源

Codeforces 286E

11.2 题目大意

先考虑给定$a_{1\sim k}$ ($k \le 10^6$)这些物品，$1 \le a_i \le m$，做一次无限背包问题。

现在给出$1 \sim m$ ($m \le 10^6$)范围内背包问题的结果，即给出所有可以凑出的数值$b_{1\sim n}$，求一组可能的$a_{1\sim k}$并最小化k，或指出无解。

11.3 算法分析

首先来判定无解。

不考虑最小化k，那么只要存在解，直接让a等于b显然会是一个合法解。那么如何判断这个解是否合法呢？稍加分析便能发现，合法当且仅当b中任意两个数的不超过m的和都已经在b中出现过。（可以用FFT 来快速判定）

如果保证了有解，试着找到k最小的那个。

先给b从小到大排序，考虑b_1，因为它是最小的，所以只能是一个物品，然后我们把b_1的所有倍数从b中删去，并把它加入物品集合。接下来找到第二小的没有删除的元素b_x，同样加入集合，然后筛去它和b_1能凑出的所有数……考虑这个算法流程，等价于维护一个物品集合，然后从小到大考虑每一个b_i，如果能被物品集合中的数凑出就跳过，否则加入到物品集合，这样得到的就是最小的物品集合。

怎么样优化这个过程呢？注意到我们的复杂度主要在于每次要对物品集合做背包来筛数，但事实上如果输入合法，b本身就是一个背包的结果了，那么一个元素究竟满足什么条件才不会被提前删去呢？我们发现对于一个i，只要不存在$j,k < i$且$b_j + b_k = b_i$，那么b_i就将加入物品集合。进一步地，$j,k < i$ 这个限制是不必要的。

所以我们用FFT求出b数组和自己的一次卷积，就得到了每个元素到底选不选做物品。

11.4 时空复杂度

时间复杂度：$O(m \log m)$

空间复杂度：$O(n+m)$

12 Theft of Blueprints

12.1 试题来源

Codeforces 332D

12.2 题目大意

给出一个n ($n \leq 2000$)个点的带权无向图，满足对于任意一个大小为k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。

12.3 算法分析

首先总方案数为$\binom{n}{k}$，所以我们只需要考虑对所有方案的权值求和。

枚举v_S，对于v_S的任意一条出边，它对答案贡献了$\binom{m-1}{k-1}$次权值（m表示v_S的出边总数）。所以v_S对所求期望的贡献为$sum * \binom{m-1}{k-1} / \binom{n}{k}$（$sum$表示$v_S$ 的出边权值和）。

直接全部加起来就是答案了。

12.4 时空复杂度

时间复杂度：$O(n^2)$

空间复杂度：$O(n^2)$

13 Rectangle And Square

13.1 试题来源

Codeforces 335D

13.2 题目大意

你有N ($N \leq 10^5$)个矩形（编号从1到N）。所有矩形的四个角都是整点，并且两组对边分别平行于X 和Y 两坐标轴。不同的矩形可能接触，但是不会重叠。（换言之，不存在一个点严格处在多个矩形的内部）

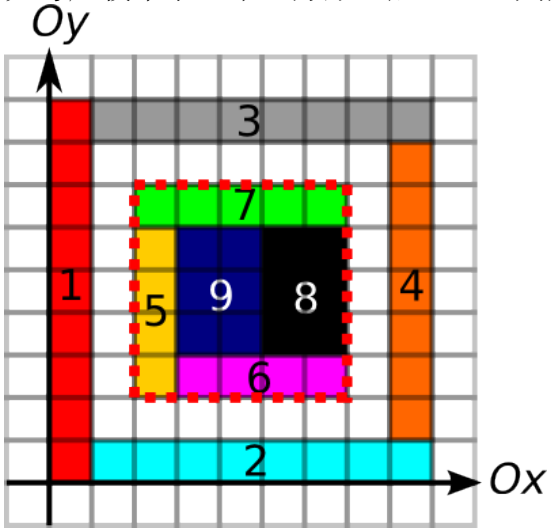
现在你的任务是：判断是否存在一个构成正方形的子集。换言之，是否存在一个矩形的子集和一个正方形，使得：

- 每个处在正方形边缘或内部的点都存在于至少一个该子集内的矩形的边缘或内部；
- 每个处在该子集内某一矩形的边缘或内部的点都处在那个正方形的边缘或内部。

坐标范围为$[0, 3000]$

13.3 算法分析

先考虑枚举了这个正方形之后，怎么判定合法。



经过一番思考我们发现有且仅有如下两个条件：

1. 正方形内部被填满了；

2. 正方形边界没有“切到”矩形。

为了快速判断，对于条件1，我们可以把每个矩形所占据的格子染色，再记录二维前缀和以便$O(1)$查询每个子矩形的和。这样如果正方形的内部和刚好等于自己的面积，则说明被填满了。

对于条件2，我们也可以看做是某一行（列）上对区间和的询问。具体来说，以行为例，观察坐标系里所有横着的小线段，那么每个矩形内部的小线段是不能被切到的，我们看做这些线段的权值为1。我们对每一行记录前缀和以便$O(1)$查询区间和，这样每次询问一条线是否“切到”了矩形就可以直接从行（列）查询区间和得到。

最后，由于拼出正方形的左上角必然是原来某个矩形的左上角，我们只需要枚举一个矩形，然后以它的左上角为正方形的左上角，再从小到大枚举边长L。每次检验合法是$O(1)$的。

13.4 时空复杂度

时间复杂度：$O(n * \max(X, Y))$

空间复杂度：$O(X * Y)$

14 Maxim and Increasing Subsequence

14.1 试题来源

Codeforces 261D

14.2 题目大意

给定数列$b_{1\sim n}$，把它连续写t次，得到一个长为$n*t$的新数列a，求a的最长上升子序列。

保证$t \le 10^9$，$n, b_{max} \le 10^5$，$n * b_{max} \le 2 * 10^7$。

14.3 算法分析

首先可以令$t = \min(t, n)$，因为超过n轮的重复是没有意义的。

考虑从前往后DP，令f_i表示最后一个数不超过i的LIS长度。那么每新加入一个数字x，就拿$f_{x-1}+1$来更新$f_{x\sim b_{max}}$，由于f单增所以更新的是一段连续的区间。

注意到$n * b_{max} \le 2 * 10^7$，也就是说，假如我们从x往右一个个更新f，遇到更新失败就直接终止更新过程。那么，我们每花费$O(1)$的复杂度，就有一个f的值增加1。由于$f_{1\sim n} \le b_{max}$，所以复杂度至多进行$n * b_{max}$次更新。

14.4 时空复杂度

时间复杂度：$O(n * b_{max})$

空间复杂度：$O(n)$

15 Cyclical Quest

15.1 试题来源

Codeforces 235C

15.2 题目大意

给定一个字符串S，接着n组询问，每组询问给定一个字符串X_i，你需要回答有多少个S的子串，满足和X_i是循环同构的。

$|S| \le 10^6$，$n \le 10^5$，$\sum_{i=1}^{n} X_i \le 10^6$。

15.3 算法分析

首先我们对原串建一个后缀自动机。

从后缀树（parent树）来考虑，对于每次询问，先把X_i倍长一份，然后枚举r，令$l = r - |X_i| + 1$，那么我们就是要维护$l \sim r$这一段插入到后缀树中的位置。这个是很好维护的，在后面插入一个字符就直接沿着SAM的边走，在前面弹出一个字符就直接修改在后缀树中的深度以及对应的节点。

注意为了不算重，如果重复走到一个节点两次，就直接跳出循环并终止计数。

15.4 时空复杂度

时间复杂度：$O(|S|)$

空间复杂度：$O(|S| + \sum X_i)$

16 Biologist

16.1 试题来源

Codeforces 311E

16.2 题目大意

有n条狗，每条狗要么是雄性要么是雌性，你可以花v_i的代价改变第i只狗的性别。

有m个人来和你打赌，对于第i个人，他会指定k_i条狗和性别x，如果那些狗的性别全变成了x，你就会获得w_i 元。这些人里有一些是你的朋友，如果不能满足他们的要求你会倒扣g 元。

求最大收益。

$n \le 10^4$，$m \le 2000$，$g \le 10^4$。

16.3 算法分析

这是个经典的网络流最小割模型。我们先收下所有奖金，再最小化扣的钱。

考虑一个点代表一条狗，定义在S割、T割分别代表着最终性别为S、T。

对于性别S的狗，将其向S连v_i的边。

对于每个人，不妨设他指定的性别是S，我们新建一个点d，d向k_i条狗连∞的边，S向d 连w_i的边，这样如果最终这些狗有在T割的，那么d也对应会在T割，这样就不得不支付w_i的代价（割）。

对于好朋友……代价多加一个g就好了。

16.4 时空复杂度

时间复杂度：$maxflow(n+km)$

空间复杂度：$O(n+km)$

17 Polygon

17.1 试题来源

Codeforces 306D

17.2 题目大意

求一个n个点的凸多边形，满足每个内角都相等，每条边两两不等。

17.3 算法分析

一个直接的想法是，定义一个$step$，然后从原点出发走n步，第i步沿着向量$(\cos((i-1)A), \sin((i-1)A))$走长度$step$（其中$A = \frac{2*\pi}{n}$），然后给$step$减少一个较小的值。

当然这样会有些小问题，有可能最后一步太过于靠右导致接不上第一步。解决办法很简单，在第$(n+1)/2+1$步（也就是多边形顶部向左走的第一步）强行走远一点就好了。

17.4 时空复杂度

时间复杂度：$O(n)$

空间复杂度：$O(n)$

18 Endless Matrix

18.1 试题来源

Codeforces 249E

18.2 题目大意

有一个形如这样的数字矩阵：

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

T ($T \le 10^4$)组询问，每次给定x_1, y_1, x_2, y_2 ($1 \le x_1, y_1, x_2, y_2 \le 10^9$)，你需要计算$\sum_{i=x_1}^{x_2} \sum_{j=y_1}^{y_2} a_{i,j}$的后10位。

注意，如果答案超过10位，你输出时需要在数字前加上“...”。

18.3 算法分析

保留后10位等价于对10^{10}取模。然而仅仅这样是不够的，我们并不知道答案是否超过10 位。解决方案是，再在模$10^{10}+9$的意义下做一次，可以大致认为答案不超过10^{10}当且仅当两次算出的答案一样。

令$F(x,y) = \sum_{i=1}^{x} \sum_{j=1}^{y} a_{i,j}$，对于每一组询问，先转化为：

$$F(x_2,y_2) - F(x_1-1,y_2) - F(x_2,y_1-1) + F(x_1-1,y_1-1)$$

现在只用考虑怎么求$F(x,y)$。

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

如图所示，绿色框框里面形如$\sum_{i=1}^{t} i = t*(t+1)/2$，可以$O(1)$算出。

对于下面row行，每一行都是连续的若干数字。求出第一行的和S_1，给答案加上$row*S_1$，这样每一行剩下没计算的值形如：

$$t$$

$$2*t+2$$

$$3*t+2+4$$

$$4*t+2+4+6$$

求和起来：

$$=\sum_{i=1}^{row} i*t+2*\sum_{i=1}^{row} i*(row-i-1)$$

这个式子拆开就是一些$\sum_{i=1}^{n} i$和$\sum_{i=1}^{n} i^2$和一些常数，都是可以$O(1)$求的。

好吧由于模数比较大得用$O(\log)$的大数乘法，但其实有一种$O(1)$的写法:

```
(a * b - (int64)((long double)a / p * b + 1e-3) * p + p) % p
```

18.4 时空复杂度

时间复杂度：$O(T)$

空间复杂度：$O(1)$

19 Greg and Caves

19.1 试题来源

Codeforces 295D

19.2 题目大意

有一个$n \times m$ $(n, m \leq 2000)$的棋盘。你需要给一些格子染黑，满足下列条件：

1. 存在区间$[L, R]$，满足L至R行每行恰有两个黑格，其他行全是白格。
2. 存在一个$t \in [L, R]$，满足对于任意$L \leq i \leq j \leq t$有$S_i \geq S_j, T_i \leq T_j$，对于任意$t \leq i \leq j \leq R$有$S_i \leq S_j, T_i \geq T_j$。

求方案数。

19.3 算法分析

我们看做是若干条黑区间叠在一起。

枚举t，即最小的满足性质2的行号。那么上面每一层都得被下一层包含，下面每一层都得被上一层包含。特别地，由于是最小的行号，上面第一层得严格被t这一层包含。

不难看出递归子结构，于是考虑DP。$f_{i,j}$表示i层最下层长j的叠放方案数；$g_{i,j}$定义相同，不过要求第二层严格包含于第一层。

$$f_{i,j} = \sum_{k=1}^{j} f_{i-1,k} * (j + 1 - k) = (j + 1) * sum1_{i-1,j} - sum_2 i - 1, j$$

其中：

$$sum1_{i,j} = \sum_{k=1}^{j} f_{i,k}$$

$$sum2_{i,j} = \sum_{k=1}^{j} f_{i,k} * k$$

g的转移类似。

进一步的我们可以求出$fs_{i,j}$表示不超过i层最下层长j的叠放方案数，$gs_{i,j}$类似。

得到了f和g，我们枚举行t，再枚举这一行区间的宽度d，给答案加上$(m+1-d)*gs_{i,d}*fs_{n+1-i,d}$。

19.4 时空复杂度

时间复杂度：$O(nm)$

空间复杂度：$O(nm)$

20 Roadside Trees

20.1 试题来源

Codeforces 264E

20.2 题目大意

Liss是一只喜欢吃坚果的松鼠。它要求你种一些坚果树。

在一条直线有n个位置可以种树，自西向东标号1 ~ n。每个月，每棵树都会长高1 米。每个月初，你都会收到一个要求。要求有两种类型：

1. 在位置p种一棵高度为h的树；
2. 砍掉从西向东数的第x棵树。（当这个位置的树被砍掉后，倒下的树会占据这个位置，之后这个位置不能再种树。）

在做完这个要求后，你需要求出当前树高的最长上升子序列。一个树的序列，当从西向东的每棵树都严格高于序列中位于它西边的树。序列的长度就是序列中树的个数。

特别的，Liss不喜欢高度相同的树，所以任何时刻树的高度是不同的。初始没有任何树。

$n \le 10^5$，$m \le 2 * 10^5$，$1 \le h_i, x_i \le 10$。

20.3 算法分析

把每棵树看做二维平面上的一个点$(pos_i, height_i)$，那么最长上升子序列，等价于：每个点只能走到自己右上方的点，走得最长的链。

定义f_i表示从点i出发走的最长的链。

我们发现，砍的树一定是最左的10棵之一，种的树一定是最下的10棵之一。那么每次操作，我们删除对应的10 棵树，或者叫10个点，然后按次序再逐个加入平面。每加入一个点，我们就要重新计算他的f 值。

由于加入时一定x或y有序，所以只要查询另一个坐标的前缀f最大值。于是我们对x 轴、y轴分别开一棵线段树来维护f值，这样每次加入一个点时，

用$O(\log W)$可以查询并计算出它的f值，再用$O(\log W)$ 就能更新线段树。（W表示坐标范围）。

20.4 时空复杂度

时间复杂度：$O(max(h_i, x_i) * m * \log W)$

空间复杂度：$O(n)$

21 Tourists

21.1 试题来源

Codeforces 286D

21.2 题目大意

直角坐标系里，有m堵墙，第i堵墙会在第t_i秒出现，覆盖直线$(0,l_i)-(0,r_i)$。

有n对游客，第i对游客在第q_i秒出发，两人分别从$(-1,0)$、$(1,0)$向y轴正方向竖直移动。你需要对于每一对游客，求出他们有多长时间无法看见彼此。

$1\le n,m\le 10^5$，$0\le l_i\le r_i\le 10^9$，$t_i\le 10^9$

21.3 算法分析

我们先对所有的l_i,r_i离散化，拆成若干单位段。然后扫描一遍，用一个堆维护当前位置覆盖着的所有线段，这样就求出了每一单位段最早被覆盖的时间。

于是我们得到了$O(m)$堵互不相交的墙，它们对答案的贡献是独立的。

我们考虑一堵墙(l,r,t)的贡献，设出发时间为s。

如果$s+r<t$，那么贡献为0。

如果$s+l>t$，那么贡献为$r-l$。

如果$t-r\le s\le t-l$，那么贡献为$s-(t-r)+1$，也就是正比于s。

想象上述的贡献写在时间轴上，一段0、一段上坡、一段常数。差分后就是一段0、一段1、一个常数、一个0、一个常数。

所以按时间回答所有游客对的询问，维护当前的答案即可。

21.4 时空复杂度

时间复杂度：$O(n\log n+m\log m)$

空间复杂度：$O(n)$

22 Little Elephant and Broken Sorting

22.1 试题来源

Codeforces 258D

22.2 题目大意

有一个$1 \sim n$的排列，m次操作，第i次交换第a_i和第b_i个位置上的数字。

现在问题来了，每次操作有50%的概率被执行，求最后逆序对个数的期望。

$n, m \leq 1000$

22.3 算法分析

令$prob_{i,j}$表示位置i上的数比位置j上的大的概率。

每次操作，我们对$prob$进行维护。

设这次操作为(a, b)，那么对应的修改为：

$$prob_{a,i}, prob_{b,i} := (prob_{a,i} + prob_{b,i})/2$$

$$prob_{i,a}, prob_{i,b} := (prob_{i,a} + prob_{i,b})/2$$

$$prob_{a,b}, prob_{b,a} := 0.5$$

最后$\sum_{i=1}^{n} \sum_{j=i+1}^{n} prob_{i,j}$即为答案。

22.4 时空复杂度

时间复杂度：$O(nm + n^2)$

空间复杂度：$O(n^2)$

23 Dima and Figure

23.1 试题来源

Codeforces 273D

23.2 题目大意

一个$n \times m$的矩阵，你需要涂黑若干格子，满足下列条件：

1. 所有黑格子构成一个四连通块，且黑格子数大于0。
2. 从一个涂黑的格子(x_1, y_1)到另一个涂黑的格子(x_2, y_2)所需的最少移动步数等于$|x_1 - x_2| + |y_1 - y_2|$。

求方案数模$10^9 + 7$。（$n, m \leq 150$）

23.3 算法分析

注意要求2，这意味着每一行、每一列黑色格子都得是连续的一段。进一步地，我们发现这样就足够了。

仔细观察我们发现，等价于从某一行开始，每一行选一个区间L_i, R_i，然后要求L先降后增，R先增后降。

于是得到状态$f_{l,r,bl,br}$表示决策完若干行，最后一行黑色格子区间为$[l, r]$，左右端点“是否下降过”的状态为bl, br。考虑新加入一行如何更新f。

先从$f_{l,r,bl,br}$转移到$f_{l+1,r,1,br}$，从$f_{l,r,bl,br}$转移到$f_{l,r-1,bl,1}$。

再从$f_{l,r,0,br}$转移到$f_{l-1,r,0,br}$，从$f_{l,r,bl,0}$转移到$f_{l,r+1,bl,0}$。

这样转移可以看做是：先决策“是否下降”，再一步步决策“是否继续缩短”或者“是否继续伸长”。

23.4 时空复杂度

时间复杂度：$O(n^2m)$

空间复杂度：$O(nm)$

24 Cow Tennis Tournament

24.1 试题来源

Codeforces 283E

24.2 题目大意

有n头牛，第i头牛的能力值为s_i，能力值互不相同。能力值强的牛可以打败能力值弱的牛。

你打算操纵比赛，有k次操纵，每次给定两个参数a_i, b_i，表示将所有两牛能力值均在$[a_i, b_i]$ 的比赛的结果取反。

求存在多少三元组(p, q, r)，使得p能打败q、q能打败r、r能打败p。

$3 \le n \le 10^5$，$0 \le k \le 10^5$。

24.3 算法分析

如果给定一张有向图求有向三元环的个数，可以先给答案加上$\binom{n}{3}$，然后减去不合法的。

不合法的情况必然有且仅有两条起点相同的边，所以枚举每个点u，给答案减去$\binom{deg_u}{2}$，最后便得到了三元环数目。所以问题关键在于求$deg_{1\sim n}$。

枚举战斗的一方u，如何求出他能战胜的牛的数目呢？首先，想象一个数组res_i表示u 和i战斗的结果（初始时$res_i = [i < u]$）。所有满足$a_i \le u \le b_i$ 的修改会产生作用，会对区间$[a_i, b_i]$ 的res取反。

于是我们用点事件，枚举u，用线段树维护res即可。操作只有区间取反、全局求和，打标记就好了。

24.4 时空复杂度

时间复杂度：$O((n+m)\log n)$

空间复杂度：$O(n+m)$

25 Sequence Transformation

25.1 试题来源

Codeforces 280E

25.2 题目大意

有一非降序列$x_{1\sim n}$，$1 \le x_1, x_n \le q$，有俩参数a、b（$a \le b$，$a(n-1) < q$）。
你要把$x_{1\sim n}$变成$y_{1\sim n}$（$1 \le y_i \le q$，$a \le y_{i+1} - y_i \le b$），变换的代价为：

$$\sum_{i=1}^{n}(y_i - x_i)^2$$

求最小的变换代价。（$n \le 300000$，$1 \le q, a, b \le 10^9$）

25.3 算法分析

令$dp_{i,p}$表示前i个数、最后一个数为p的最小代价。则有：

$$dp_{i,p} = (p - x_i)^2 + \min\{dp_{i-1,q} \mid (p - b \le q \le p - a)\}$$

由于p可能是小数，不妨直接把dp_i看做一个关于p的函数。
$dp_1(p) = (p - x_1)^2$。
$dp_1'(p) = 2(p - x_1)$，可以看出这个导函数是严格单增的。
下面，假设我们已经证明了dp_i'是严格单增的，并且$\min dp_i = dp_i(k)$，$dp_i'(k) = 0$。
设$dp_{i+1}(p) = (p - x_{i+1})^2 + f_{i+1}(p)$，则有：

$$f[i+1](p) = \begin{cases} dp[i](p-a), & p < k + a \\ dp[i](p-b), & p > k + b \\ dp[i](k), & k + a \le p \le k + b \end{cases}$$

$$f[i+1]'(p) = \begin{cases} dp[i]'(p-a), & p < k + a \\ dp[i]'(p-b), & p > k + b \\ 0, & k + a \le p \le k + b \end{cases}$$

注意到对于导函数，我们实际上就是把dp'_i从$p = k$处切开，然后左边右移a个单位，右边右移b 个单位，$[k+a, k+b]$填充0。这样就得到了f'_{i+1}，一个非降函数。再加上$2(p - x_{i+1})$这个严格增函数，得到的依然是一个严格增函数。

于是我们维护这个导函数，每次找到零点，切开，位移。用平衡树维护的话是$O(n \log n)$ 的。

25.4 时空复杂度

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

26 Olya and Graph

26.1 试题来源

Codeforces 305D

26.2 题目大意

Olya现在有一张n个点、m条边的有向无权图。现在我们以某种方式给所有点从1到n 标号，从而保证原图中任意从u到v的有向边满足不等式：$u < v$。

现在Olya想知道有多少种方案添加任意数量（可能是0）的有向边，使得该图满足下列条件：

1. 从点i出发，可以到达点$i+1$，$i+2$,...,n。
2. 任意从u到v的有向边满足不等式：$u < v$。
3. 两点之间最多有一条边。
4. 对于一对点i、j（$i < j$），若$j-i \le k$，那么从i到j的最短距离等于$j-i$条边。
5. 对于一对点i、j（$i < j$），若$j-i > k$，那么从i到j的最短距离等于$j-i$或$j-i-k$条边。

我们认为两种添加边的方案不同，当且仅当存在至少一对点i、j(i¡j)，第一张图中有一条从i连向j的边，而第二张图中没有。

帮助Olya。由于要求的答案可能太大，将答案对10^9+7取模后输出。

$n, k \le 10^6$，$m \le 10^5$

26.3 算法分析

仔细分析题中描述的性质便能发现，这种图是一条链，i连向$i+1$，然后某些点i可能有一条连向$i+k+1$ 的边（我们称这种点为黑点）。

等价于，一排n个点，每个点颜色要么黑要么白，你要再染黑若干点，满足最远的黑点对距离不超过k。

枚举最右的黑点的位置，得到黑点们的左界，然后用前缀和统计出中间有多少白点，给答案加上2^{white}。

26.4 时空复杂度

时间复杂度：$O(n+m)$

空间复杂度：$O(n+m)$

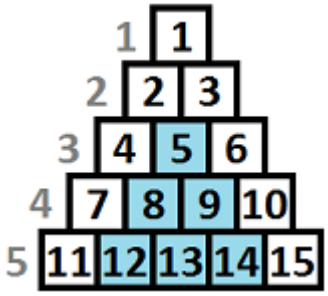
27 Transferring Pyramid

27.1 试题来源

Codeforces 354D

27.2 题目大意

有一个如图的金字塔，一共n行，第i行有i格。



你可以花费3的代价，覆盖单个格子。

你也可以花费$2 + size$的代价，覆盖一个子金字塔（比如图中蓝色部分）。

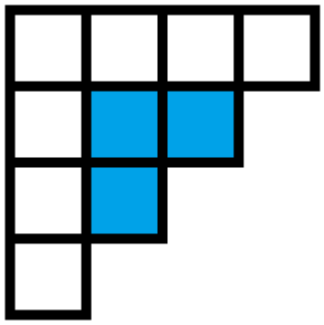
你需要用最少的代价覆盖指定的k个格子。

27.3 算法分析

初始时我们先让k个格子都花费3的代价被单点覆盖，然后问题变成了，有k个格子价值3，你可以花费$2 + size$的代价来覆盖一个子金字塔，最大化收益（价值和减去代价）。

由于总价值只有$3 * k$，而覆盖一个子金字塔的代价是$O(h^2)$的（h表示子金字塔高度），所以h是$O(\sqrt{k})$的。

我们把金字塔转一下，变成这个样子：



然后就可以一行一行的DP了。

令$f_{i,j}$表示，考虑完了前i行，当前后缀j是被覆盖着的，此时的最大收益。

转移方程：

$$f_{i,j} = \max\{f_{i-1,j}, f_{i-1,k|(k>j)} - cost(j)\} + suffixValueSum_{i,j}$$

记录后缀最大值来做到$O(1)$转移，用滚动数组优化空间。

27.4 时空复杂度

时间复杂度：$O(n * \sqrt{k})$

空间复杂度：$O(n)$

28 Maxim and Calculator

28.1 试题来源

Codeforces 261E

28.2 题目大意

你有两个变量a、b，初始时$a = 1$、$b = 0$。

每次操作你可以：

1. $b+ = 1$
2. $a* = b$

求存在多少个$x \in [l, r]$满足存在一种操作方式，使得至多操作p步之后a的值为x。

$2 \le l \le r \le 10^9$，$p \le 100$

28.3 算法分析

由于$p \le 100$，那么$b \le 100$。由$[1, 100]$中的数字乘出来的不超过10^9的数，经过搜索发现并不是特别多，只有约$T = 3000000$个，排序后，不妨设第i个叫num_i。

所以我们枚举b最后的值，然后维护f_i表示使得$a = num_i$所需要的最少步数。更新时只需要从左到右扫一遍，枚举i然后维护最小的j满足$num_j >= num_i * b$，判断一下i是否能转移给j就好了。

更新后所有$f_i + b \le p$ and $L \le num_i \le R$的都是符合要求的。由于T不大，开一个标记数组记录每个数字是否被统计过即可。符合要求就统计进答案。

28.4 时空复杂度

时间复杂度：$O(Tp)$

空间复杂度：$O(T)$

29 Printer

29.1 试题来源

Codeforces 253E

29.2 题目大意

我们考虑一个有这样功能的网络打印机：他从时刻0开始工作，每一秒钟他打印一页纸。某些时刻他会收到一些打印任务。我们知道打印机会收到n个任务，我们将它们分别编号为连续的整数$1 \sim n$，并且第i个任务用三个参数描述：t_i表示接到的时间，s_i表示任务要求你打印多少张，以及p_i表示任务的一个优先级。所有任务的优先级互不相同。

当一个打印机收到一个任务时，任务会进入一个队列并留下直到完成了这个任务为止。在任务队列非空时，每个时刻，打印机会选择队列里优先级最高的一个任务，打印一页。你可以想象任务进入队列是瞬间的事情，所以他可以在收到某个任务的时刻去执行这个任务。

你会得到除了某个任务以外所有任务的信息：你不知道某个任务的优先级是多少。然而，我们还额外的知道这个任务他完成时的时刻。我们给你这些信息，请求出这个未知的优先级并对每个任务输出它完成时的时刻。

$n \le 50000$，$t_i, s_i, p_i \le 10^9$。

29.3 算法分析

注意到同一个任务，其他参数不变，优先级越高，打印完成的时间至少不会更晚。

于是我们二分这个未知的优先级，然后直接模拟一遍，用一个堆存下所有的任务，把事件点按时间排序逐个加入，就能得到此时所求任务的完成时刻。

29.4 时空复杂度

时间复杂度：$O(n \log^2 n)$

空间复杂度：$O(n)$

30 Sheep

30.1 试题来源

Codeforces 309E

30.2 题目大意

有n ($n \leq 2000$)个元素，第i个元素有两个属性l_i, r_i，代表一个区间。两个元素之间有一条边，当且仅当它们的区间有交。

求一个这些元素排列，使得有边的元素间的最远距离最小。

30.3 算法分析

先二分答案D，问题变为限制每两个有边的元素距离不超过D，求是否可行。

考虑从左到右依次放下每个元素，令mx_i表示元素i最晚在什么位置必须放下，初始时$mx_{1\sim n} = \infty$，每放下一个元素后把与这个元素有边的、还未放下的元素的mx 修改一下。

现在的问题是如何决定当前位置放下的元素。令$need_i$表示有多少个j满足$mx_j \leq i$，如果存在$need_i > i - curPos + 1$则说明无解。否则我们找到最小的满足$need_i = i - curPos + 1$的i，然后在mx不超过i的未放置中选一个放下即可。

30.4 时空复杂度

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

31 Summer Homework

31.1 试题来源

Codeforces 316E3

31.2 题目大意

有一个数列$a_{1\sim n}$，m次操作，每个操作有三种类型：

1. 将a_x赋值成y。
2. 求和$\sum_{x=0}^{r_i-l_i}(fib_x * a_{l_i+x})$。
3. 将$a_{l\sim r}$加上d。

所有操作在模10^9的意义下进行。（$n, m \leq 200000$）

31.3 算法分析

注意$fib_i = fib_{i-1} + fib_{i-2}$这个递推式，令$F(x)$表示前两个数为0、$x$的$fib$ 数列，那么显然有$F(x) + F(y) = F(x + y)$、$F(x * k) = F(x) * k$。

这样我们用一棵线段树来维护数列，每个节点$[l, r]$存着$\sum_{i=0}^{r-l}(fib_i * a_{l+i})$。信息合并的时候用上一个自然段的式子“抬高”右子树的值，和左子树相加。

至于单点赋值和区间加值，线段树上打标记就可以轻松惬意地解决了。

31.4 时空复杂度

时间复杂度：$O(m \log n)$

空间复杂度：$O(n)$

32 Colorado Potato Beetle

32.1 试题来源

Codeforces 243C

32.2 题目大意

从前有一片$(10^{10}+1)\times(10^{10}+1)$的农田，分成了单位正方形小块。

有一台农药车，一开始在最中间的那个格子里，然后移动n回合，每回合会朝着上下左右的某个方向行驶整数个格子x_i。如果某个格子被农药车至少到过一次，那么会被洒上农药。

现在来了一批蝗虫，它们会从边界开始，往内部沿着没有撒农药的格子floodfill。求有多少格子能幸免于虫灾。

($n\le 1000$，$x_i\le 10^6$)

32.3 算法分析

由于地图太大了，为了方便地模拟floodfill，不妨考虑对地图进行离散化。

对于农药车走到的$n+1$个关键点a_i,b_i，我们把a_i-1,a_i,a_i+1加入“关键的x”，b_i-1,b_i,b_i+1加入”关键的y”，然后按关键坐标分割坐标系。那么如此切割之后地图被分成了$O(n^2)$个矩形，每个矩形区域要么全免于虫灾，要么全不免。

所以离散化之后，先模拟农药车在离散化后的地图里跑一遍，路过的格子染成黑色，最后直接从边界开始做floodfill，把没有被fill的格子面积加起来就是答案。

32.4 时空复杂度

时间复杂度：$O(n^2)$

空间复杂度：$O(n^2)$

33 Good Substrings

33.1 试题来源

Codeforces 316G3

33.2 题目大意

求字符串s有多少个本质不同的子串是好的。

一个字符串t是好的，当且仅当它满足全部下列n条规则。

每个规则形如，t在p_i中出现了l_i到r_i次。

$|s|, |p_i| \leq 50000$，$0 \leq n \leq 10$。

33.3 算法分析

把s和$p_{1\sim n}$用特殊字符隔开连成一个字符串，然后建立SAM，也得到了前缀树。

众所周知，a在b中的出现次数，等价于拿a在b的SAM里走一趟，所在前缀树节点的叶子个数。

那么同样的，我们在这棵合并了的前缀树上，求出每个节点子树内，每种类型的叶子的个数。然后枚举前缀树每一条树边，如果底下那个点符合要求，就把这条边上的字符个数加进答案。

33.4 时空复杂度

时间复杂度：$O(nL)$

空间复杂度：$O(nL)$

34 Tree and table

34.1 试题来源

Codeforces 251E

34.2 题目大意

小佩蒂亚非常喜欢树。最近她的母亲送了他一棵有$2n$个节点的树，佩蒂亚立即决定把这棵树放在一个2行n列的矩形表格中，并满足以下条件：

1. 表格的每个格子恰好放入一个树上的节点，每个树上的节点恰好放入一个表格的格子。
2. 如果两个树上的节点间有一条边相连，那么它们放入的格子有一条公共边。

现在佩蒂亚想知道，有多少种把树放进表格的方法。

两种方法是不同的当且仅当存在一个节点放入的格子不同。

因为太大的数字让佩蒂亚感到恐惧，只需要输出答案对10^9+7取模后的值即可。

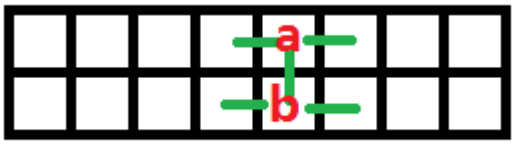
$n \le 10^6$

34.3 算法分析

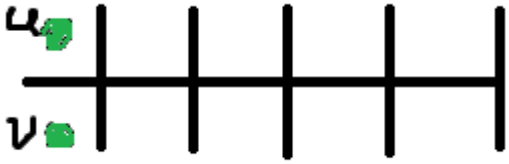
如果某个点度数大于3，说明一定无解。

否则，如果所有没有度数为3的点，也就是说是一条链了。这样的方案数可以用递推来求，f_n表示n 列时从左上角出发的答案，枚举第一步怎么走，显然有$f(n) = f(n-1)+1$，即$f(n) = n$。回到原问题，枚举出发点，得到计算式：$\sum_{i=1}^{n}(2*(i-1+n-i))+4 = 2n^2-2n+4$。

下面开始考虑，如果存在某个点度数为3该怎么办。我们随便找一个度数为3的点a，把a 作为根，然后枚举一个将要和它同列儿子b，再枚举a、b的其他儿子的顺序，这样棋盘便被分成了两半。



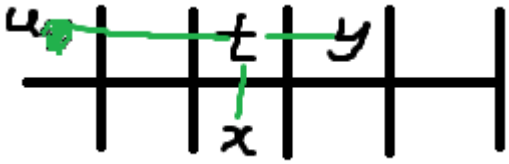
注意此时a的所在列是可以由左边的子树$size$唯一确定的。由于两边已经被完全分开，只需对一边进行讨论。这里以右边为例。



如果u、v都还各有一个儿子，就都往右搭一个儿子，然后这俩儿子变成了一个递归子问题。

如果某个时刻u或v的度数大于1了，说明无解了。

否则，只有一个点有一个儿子，不妨设那个点是u。我们把u的儿子搭到右边，然后递归下去，问题变成了这样子：



如果此时的u儿子数为2，枚举一下它的俩儿子顺序，又同样地继续递归下去了。

否则，此时u的儿子数为1，我们找到u下面第一个儿子数大于1的节点t（如果不存在说明是一条链，直接算出方案数），t的儿子数得是2，设x、y是t的俩儿子，注意到一方面我们要选一个儿子（比如x）来填充左下角的链状空隙，另一方面右边形成了一个新的递归子问题。用同样的办法，枚举儿子顺序，逐个讨论可行性，最终都能转移到右边新的递归子问题。

至此，我们发现，当确定了最初的根a以后，这个记忆化搜索的状态量是$O(n)$的。我们需要通过一些预处理，进行精细的实现，最终可以做到$O(n)$的复杂度。

34.4 时空复杂度

时间复杂度：$O(n)$

空间复杂度：$O(n)$

35 Liars and Serge

35.1 试题来源

Codeforces 256D

35.2 题目大意

给你n、k，求有多少个数列$a_{1\sim n}$，满足：

- $1 \le a_i \le n$
- 恰有k个数值x满足x在a中出现次数并非x

答案对777777777取模。（$1 \le k \le n \le 2^8$，保证n是2的整次幂）

35.3 算法分析

定义一个数值x是好的，当且仅当它在a中出现了x次。

我们来DP：考虑按数值从1到n来一组组放，每放一组的时候就在之前的里面插空。这样的好处是，可以方便地维护好数值的个数。

$f_{i,l,g}$ 表示考虑完数字$1\sim i$、选了l 个数字出来、有g个好数值的方案数。

枚举当前数值i放c个，简单的转移：

$$f_{i,l+c,g+[c==i]} += f_{i-1,l,g} * \binom{l+c}{c}$$

什么，好像还是过不了？题目中说n是2的整次幂……

难道有什么好的性质吗？

……

……

原来可以打表!

35.4 时空复杂度

时间复杂度：$O(n^4)$

空间复杂度：$O(n^3)$

36 Rotatable Number

36.1 试题来源

Codeforces 303D

36.2 题目大意

Bike是个十分喜欢数学的聪明孩子。他发明了“可旋转数”，其灵感来自于142857。

正如你所见，142857是一个十分神奇的数，因为所有从它通过旋转得到的数都是它自己乘以1,2,3...,6 （从1 到数的长度）。旋转一个数就是将它的最后一位数字放到最前面。比如说，通过旋转12345你能够得到这些数：12345,51234,45123,34512,23451。值得一提的是这里允许有前导0。因而4500123、0123450都能够通过旋转0012345得到。你可以看看142857满足条件的原因了。下面的6个方程都在十进制下成立：

142857 * 1 = 142857;

142857 * 2 = 285714;

142857 * 3 = 428571;

142857 * 4 = 571428;

142857 * 5 = 714285;

142857 * 6 = 857142;

现在，Bike提出了一个问题。他将“可旋转数”推广到了任意进制b。如上所示，142857是十进制下的一个“可旋转数”。另外一个例子是二进制下的0011。下面的4个方程都在二进制下成立：

0011 * 1 = 0011;

0011 * 10 = 0110;

0011 * 11 = 1001;

0011 * 100 = 1100;

他想要找到最大的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）。

$n \leq 5 * 10^6$，$x \leq 10^9$。

36.3 算法分析

不妨先在10进制下考虑。

以142857为例，考虑1/7的除法过程，每次*10，然后除以7，如果两次的余数相同，就开始循环了。所以为了保证循环节长度为n，需要满足$10^{1\sim n} \bmod (n+1)$互不相同。

所以b满足条件，当且仅当b是模$n+1$意义下原根。

我们求出$n+1$的所有约数，然后从大到小枚举b，用快速幂对n的每个约数d判断$b^d = 1 \mod (n+1)$是否成立，找到第一个符合条件的b就跳出循环。

36.4 时空复杂度

时间复杂度：$O(\sqrt{n}\log n)$

空间复杂度：$O(\sqrt{n})$

37 GCD Table

37.1 试题来源

Codeforces 338D

37.2 题目大意

有一个n行m列的表格G，$G_{i,j} = \gcd(i, j)$。

给你一个正整数数列$a_{1\sim k}$，询问这个数列是否在表格的某一行出现过。

$n, m, a_i \leq 10^{12}$，$k \leq 10^5$

37.3 算法分析

首先，第i行的每个数显然都是i的约数。令$h = \mathrm{lcm}_{i=1}^{n} a_i$，那么数列$a$的出现行号必然是$h$的倍数。进一步地我们发现，如果在第$kh$行出现了，由于是数字的意义是最大公约数，那么搬到第h行来，也没有理由不出现。于是我们确定了要出现就要在第h 行出现。

接下来考虑怎么确定列呢。类似地，我们挖掘列的性质。不妨设a_1在第l列出现，容易得到下列式子：

$$l = 0 \mod a_1$$

$$l + 1 = 0 \mod a_2$$

$$\cdots$$

$$l + (n - 1) = 0 \mod a_n$$

用中国剩余定理可以解得一个最小的可行的l，其余可行的l必然要加上若干倍的h，即a的lcm。思考一下发现和之前确定行同样的理由，要出现肯定a_1对应l是最好的了。

得到了h和l，直接算出第h行的从l开始连续n个本来是啥，然后比较一下是否和a符合，就知道该输出YES还是NO了。

37.4 时空复杂度

时间复杂度：$O(k \log n)$

空间复杂度：$O(k)$

38 More Queries to Array...

38.1 试题来源

Codeforces 266E

38.2 题目大意

你有一个数列$a_{1\sim n}$，m个操作，每个操作是如下两种类型：

1. 把$a_{l\sim r}$赋值为k
2. 求和$\sum_{i=l}^{r} a_i * (i-l+1)^k$，答案对$10^9+7$取模。

$n \le 10^5$，$a_i, x \le 10^9$，$k \le 5$。

38.3 算法分析

一看就是一道简单的线段树练习题。

$$\sum_{i=l}^{r} a_i * (i-l+1)^k$$

$$= \sum_{i=l}^{r} a_i * (\sum_{t=0}^{k} \binom{k}{t} * (i+1)^t * (-l)^{k-t})$$

$$= \sum_{t=0}^{k} \binom{k}{t} * (-l)^{k-t} * (\sum_{i=l}^{r} a_i * (i+1)^t)$$

所以只要对于每个t分别维护$a_i * (i+1)^t$就好了，也就是t棵线段树。
修改的时候对于每个t都改一下，询问的时候用上面的计算式枚举t来求和。

38.4 时空复杂度

时间复杂度：$O(mk \log n)$
空间复杂度：$O(n)$

39 Binary Key

39.1 试题来源

Codeforces 332E

39.2 题目大意

考虑这样一种加密方式。p是原串，q是钥匙串（仅由0、1组成）。

对于所有i满足$q_{i \bmod |q|} = 1$，把p_i依次塞到加密串s的右侧。

现在给定p和s，你需要求一个字典序最小的长为k的钥匙串。

$|p| \le 10^6$，$|s| \le 200$，$k \le 2000$。

39.3 算法分析

我们发现，为了让字典序最小，我们不可避免地需要逐位确定s，而逐位确定为了知道当前这一位对应p的哪些地方，我们得知道s一共有多少个1。

所以，先枚举s中1的个数c，然后从右往左，能填1就填1。检查方法是，暴力在p和s中对应的位置跳，检查对应字符是否都符合。

39.4 时空复杂度

时间复杂度：$O(|s|^2k)$

空间复杂度：$O(|p|)$

40 Deja Vu

40.1 试题来源

Codeforces 331E2

40.2 题目大意

给一张n个点m条边的无重边有向图，每个点有标号，每条边上写着若干个标号。

求存在多少条路径，满足这条路径依次经过的点的标号和依次经过的边的那些标号完全一致。答案对10^9+7取模。

$n \le 50$，$m \le n*(n-1)/2$。

设k为边上的数字总数，$k \le 10^5$。

40.3 算法分析

定义E表示[幻觉数]-[现实数]，那么运动过程是这样子的：

起初：$E=-1$。因为就一个起点。

接着：E在负值内波动。这段时间的幻觉进度始终小于现实进度。

然后：E突然变成非负值。这时经过了一条转折的边。

最后：E变成0。E非负以后现实就一直在追赶幻觉。

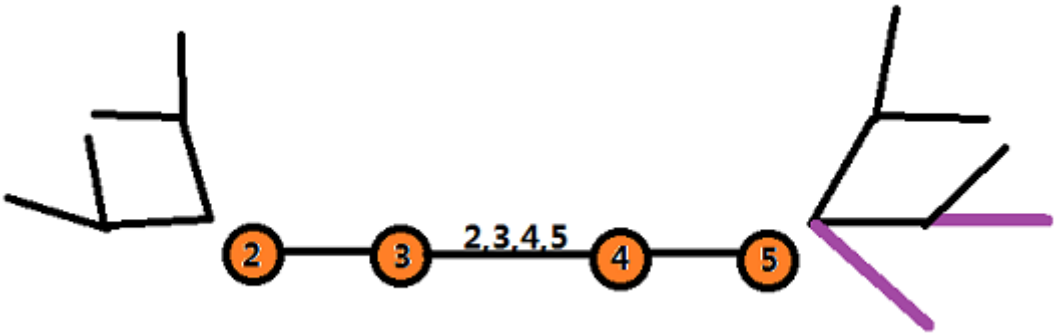
当然，E变成0之后还有可能：

再走一条非空的边，进行一次“受迫运动”；或者是经过一条空的边，再递归来一次运动。

路径的计数，自然考虑按起点分类。

不过直接从一个起点出发，完全不知道往哪走。似乎转折边比较好找——这条边的两端点包含于边上的序列。

找到转折边后，我们以之为第一条转折边来统计：



路径大概是这样的。橘色的是从转折边往两边受迫运动出的路径，紫色的是递归的新的路径。两侧发散状的东西表示受迫运动增广出的路径。（注意到左边没有紫色，计数就不会重）

这样拿出所有的单位路径之后，就得到了所有状态的转移边。我们给每个状态先标号，然后标号之间连转移边。最后来一遍拓扑排序，求出DP值。

40.4 时空复杂度

时间复杂度：$O(n^4 + k)$

空间复杂度：$O(n^3 + k)$

41 Xenia and String Problem

41.1 试题来源

Codeforces 356E

41.2 题目大意

给你一个仅有小写英文字母的字符串t，你可以修改至多一个位置的字符，使得t的美丽值最大。

一个串p的美丽值为p中所有Gray子串的长度的平方和。

一个串s被称为是Gray串，当且仅当：

1. 串长$|s|$是奇数
2. 字符$s_{(|s|+1)/2}$在串s中只出现了一次
3. $|s| = 1$或者子串$s_{1\sim(|s|+1)/2-1}$和$s_{(|s|+1)/2+1\sim|s|}$是相同的Gray串

41.3 算法分析

观察Gray串的定义，我们发现它是递归定义的，由于每次合并小Gray串之后长度会加倍，因此可以分成$O(\log n)$层的Gary串。

我们递推求出$cnt_{i,c}$表示$t_{1\sim i}$中字符c的出现次数，然后使用字符串哈希，就能轻松求出$gray_{i,lv}$表示以i为中心是否有第lv层的Gray串。

可是题中允许修改至多一个位置，我们不妨试着求出$get_{i,c}$表示把t_i修改成字符c 的美丽值的收益。

考虑下列几种情况即可。

- 破坏了某个Gray串（又分为从中点破坏、从旁边破坏）
- 新增出了某个Gray串

这些情况的贡献都在求Gray串的同时顺便讨论一下就行了。

41.4 时空复杂度

时间复杂度：$O(n\log^2 n)$

空间复杂度：$O(n\log n)$

42 Dima and Game

42.1 试题来源

Codeforces 273E

42.2 题目大意

有n条线段$[l_i, r_i]$，满足$1 \le l_i \le r_i \le p$。

两位玩家轮流行动，每次可以进行三种操作中的一种：

1. 将某一条线段$[l, r]$替换为$[l + \lfloor \frac{r-l}{3} \rfloor, l + 2 * \lfloor \frac{r-l}{3} \rfloor]$
2. 将某一条线段$[l, r]$替换为$[l, r - \lfloor \frac{r-l}{3} \rfloor]$

求有多少组可能的线段，使得先手必胜。答案模$10^9 + 7$。

$n \le 1000$，$p \le 10^9$。

42.3 算法分析

首先注意到一根线段我们只关心其长度，一根长度为l的线段有$p + 1 - l$种放置方法。

博弈问题直接上SG函数，考虑f_i表示长度为i的线段的SG值，那么有$f_i = \text{mex}(f_{\lfloor \frac{i}{3} \rfloor}, f_{i-\lfloor \frac{i}{3} \rfloor})$。暴力转移显然是不行的，但是注意到$f$的值域是$[0, 2]$，也就是说会有大量相同的段。我们把相同的段合并起来，然后每次，考虑$x/3$和$x - x/3$这两个指针，如果指向的段都不变，那么求出来的f值也不会变，于是每次可以求出一大批的f作为新的一段。

打表后发现，这样做效果很好，能很快统计出cnt_x表示SG值为x的放置方案数。

求完了cnt之后，直接$dp_{i,x}$表示选了i个异或值为x的方案数，然后DP转移就行了。

42.4 时空复杂度

时间复杂度：$O(n + p)$

空间复杂度：$O(n + p)$

43 Lucky Tickets

43.1 试题来源

Codeforces 333C

43.2 题目大意

给定k、m，你需要找出m个8位数（允许前导0），满足每个数都可以通过插入一些运算符号（”+”、”-”、”*”、”/”）和括号后最终结果等于k。

43.3 算法分析

可以通过类似搜索+构造得到。

考虑后4位，我们枚举后4位是多少，然后搜索通过插入运算符这些数可以算出那些值，如果某个值和k 的差值在0 ~ 9999，那么直接在前4位补上对应的数字，就得到了一个合法的8位数。

我们发现这样构造出来的方案是绰绰有余的。

43.4 时空复杂度

时间复杂度：$O(1000 * 3^4)$

空间复杂度：$O(k)$

44 Torcoder

44.1 试题来源

Codeforces 240F

44.2 题目大意

给一个长n的字符串s，以及m个操作。

每次操作给定l, r，然后会把$s_{l\sim r}$重排列成一个字典序最小的回文串（如果没有办法完成则不操作）。

你只需要求出m次操作后的串长什么样。

$n, m \leq 10^5$

44.3 算法分析

注意一个操作，我们统计出cnt_c表示字符c出现了多少次。

如果有多于1个字符出现次数为奇数，说明无解。

否则如果有1个出现次数为奇数的字符，那么正中间必然就是它。接下来为了使字典序最小，我们按从$a\sim z$的顺序依次填入每种字符，即最左放一半，最右放一半。

考虑上述操作如何快速实现，我们只需要一棵线段树来统计区间内字符出现次数即可。

44.4 时空复杂度

时间复杂度：$O(26*n\log n)$

空间复杂度：$O(26*n)$

45 Playing with String

45.1 试题来源

Codeforces 305E

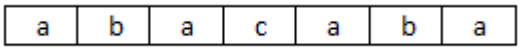
45.2 题目大意

两个小哥正在玩一个游戏。

两个小哥轮流行动，不能操作的人输。

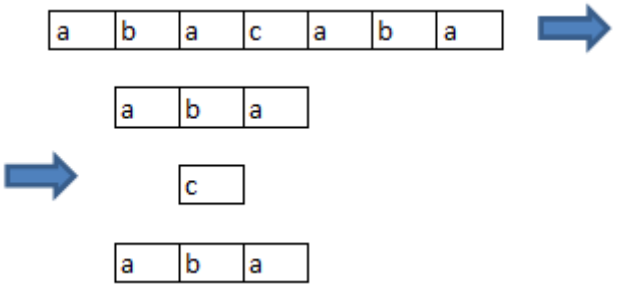
游戏开始前裁判买老师会在方格纸上写下一个字符串，每个格子包含一个字母。

比如字符串”abacaba”长这样：



一个小哥的操作分这么几步：

1. 这个小哥选择一张纸，我们称上面写着的字符串为t。注意一开始时只有一张可选纸。
2. 这个小哥选择一个i $(1 \le i \le |t|)$使得存在一个正整数k $(0 < i-k, i+k \le |t|)$，满足$t_{i-1} = t_{i+1}$ and $t_{i-2} = t_{i+2} ... t_{i-k} = t_{i+k}$
3. 这个小哥以把这张纸的第i个字母的两侧撕开使得这张纸分成3份:$t_{1\sim i-1}, t_{i\sim i}, t_{i+1\sim |t|}$如下图：



假如两个小哥都身经百战，使用最优策略，你需要确定胜者是谁。假如先行动的人胜，那么你还需要输出可以帮助这个小哥获胜的第一步行动选择的位置。如果有多个输出最小的那个。

$|s| \le 5000$

45.3 算法分析

每次切割的位置，等价于需要满足左边的字符等于右边的字符。

那么这道题等价于，一个序列一开始全是白色，两人轮流操作，每次染黑一个白格子，并且不能有两个相邻的格子被同时染黑，问先手是否必胜以及第一步该如何选。

f_i表示长为i的序列的答案，通过SG定理我们可以很方便地得到转移：

$$f_i = \mathrm{mex}(f_{i-2},\ \mathrm{mex}_{j=2}^{i-1} f_{j-2} \oplus f_{i-(j+1)})$$

然后我们求出原序列哪些格子可以染黑，分成若干个子问题，每个求出SG值，然后异或起来得到总问题的答案。

最后枚举一下第一步的决策，很容易得到剩下的SG值，判断一下选择最小就好了。

45.4 时空复杂度

时间复杂度：$O(|s|^2)$

空间复杂度：$O(|s|)$

46 Graph Game

46.1 试题来源

Codeforces 235D

46.2 题目大意

求点分治一个n个点的无向基环外向树的复杂度期望。

$n \leq 3000$

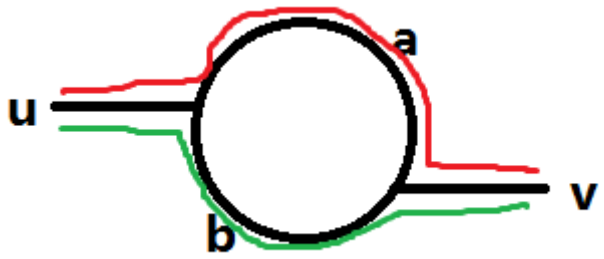
46.3 算法分析

方便起见我们先考虑树该怎么做。

定义$P(i,j)$表示删除i的时候i和j还连通着的概率，那么所求的期望$=\sum_{i=1}^{n}\sum_{j=1}^{n}P(i,j)$。考虑$P(i,j)$，意味着$i$必须是链$i-j$中第一个被删掉的点，概率为$\frac{1}{dist_{i,j}}$，所以求和就好了。

现在来拓展到基环外向树。

同样考虑$P(u,v)$，如果u,v在同一棵子树，那么计算式和树的情况相同。否则，一定是下图的形式：



从u到v有两条路径a、b，那么$P(u,v)=\frac{1}{|a|}+\frac{1}{|b|}-\frac{1}{tot}$，即先加上”是链$a$第一个删的概率”、“是链$b$第一个删的概率”，再减去同时是链$a$、$b$第一个删的概率。

所以我们记录一下环长，枚举u，然后dfs每个v，就能计算$P(u,v)$了，最后全部加起来即可。

46.4 时空复杂度

时间复杂度：$O(n^2)$

空间复杂度：$O(n)$

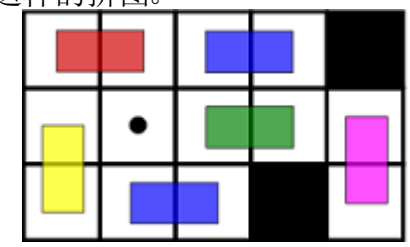
47 Xenia and Dominoes

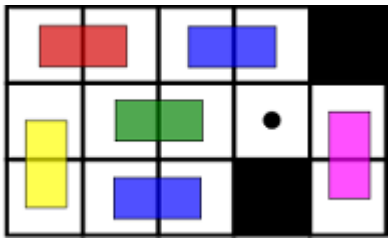
47.1 试题来源

Codeforces 342D

47.2 题目大意

Xenia非常喜欢拼图。她特别喜欢由多米诺骨牌组成的拼图。图片中就是一个这样的拼图。





一个拼图是一个$3 \times n$的桌子，除掉一些禁止块(forbidden cells)（图中的黑色正方形），并包含多米诺骨牌。一个拼图被称作合法当它符合以下条件：

每个多米诺骨牌覆盖正好两个非禁止块；

没有两个多米诺骨牌覆盖桌子上的同一块区域；

有且仅有一个非禁止块没被任何多米诺骨牌覆盖（图中的圆点）。

要完成一个拼图，你需要用若干步把空格子从开始位置移动某个指定位置。一步移动是在保证拼图合法的情况下，把一个多米诺骨牌移到空格子里。横向的多米诺骨牌只能横向移动，纵向的多米诺骨牌只能纵向移动。你不能旋转多米诺骨牌。图中表示了一个合法的移动。

Xenia有一个$3 \times n$的带禁止块和一个圆圈标记(circle-marked)的格子的桌子。Xenia 还有很多完全一样的多米诺骨牌。现在Xenia想知道，如果她把多米诺骨牌放在桌子上，能有多少种不同的合法的拼图。同时，Xenia要求圆圈标记的格子没有被覆盖。这个拼图还必须至少能移动一次。

帮助Xenia统计上述拼图的种数。这个数字可能很大，输出它对$10^9 + 7$取模的余数。

$n \leq 10^4$

47.3 算法分析

简单的状压DP。

注意到行只有3，那么我们按列转移，$f_{j,s}$表示考虑完了前j列，多米诺骨牌轮廓线上的状态为s（有没有插头），此时的方案数。

然后枚举下一列怎么拼就可以转移了，注意题目要求一定要至少能移动一步，也就是圆点的四周有一个骨牌指向圆点，特判一下就行了。

47.4 时空复杂度

时间复杂度：$O(n*(2^{row})^2)$

空间复杂度：$O(n*2^{row})$

48 Ping-Pong

48.1 试题来源

Codeforces 319E

48.2 题目大意

在这个问题中，每个时刻您都有一个区间的集合。您每次可以从集合中的区间$[a,b]$移动到另一个满足$c<a<d$或者$c<b<d$的区间(c,d)。您需要判断是否有一种从区间x到区间y的移动方案。

一共有n个操作，每个操作要么加入一个区间，要么询问是否有路径。

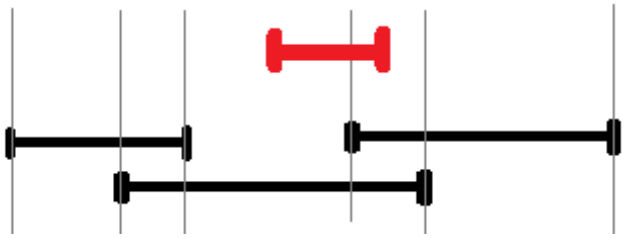
保证加入的区间长度严格递增，$n\le 10^5$。

48.3 算法分析

分析可知，两个相交的区间之间有一条无向边，两个包含的区间之间有一条有向边。

我们把题中的路径倒过来，变成每次要么走到一个相交的区间，要么走到一个自己包含着的区间。

对于无向边，我们用并查集维护。考虑怎么维护“走到一个自己包含着的区间”。



比如红色的终点，这些黑色线段是起点所在并查集里的线段。

首先可以证明，每次走进自己包含的区间，那么新到的并查集的范围会被原来的并查集的范围给包含。因为假设走进自己包含的区间t，t被之前的f包含，那么不论t如何利用相交移动，为了不和f相交，是无法突破f的左右边界的。

也就是说“红色线段在起点并查集的范围内”是一个必要条件。

下面证明这个条件是充足的。

红色线段一定被某条黑色线段所包含。否则，每条黑色线段和红色线段的关系只有“相离”、“被包含”。这样一来，由于黑色线段的范围包含了红色线段，说明两侧都有黑色线段，并且两侧和红色线段都“相离”，也就是被隔开了。那么这说明这些黑色线段不属于同一并查集。

下面考虑怎么实现这个算法。

为了对相交关系维护并查集，在插入一个新的线段后，我们需要查询这个线段和之前哪些线段相交。由于长度递增，那么只要查询新线段的l和r被哪些线段覆盖即可。这个可以用线段树维护，每个节点存着当前区间有哪些线段覆盖着。

考虑复杂度。查询时单点询问，把线段树上的祖先链中的全部标号拿出来，并查集并起来。然后把这个并查集的范围作为一条新线段，给$O(\log n)$个区间里塞进一个线段的标号。（因为新区间某个端点落在这个并查集范围内就一定和某条相交）

坐标范围较大，需要离散化。

48.4 时空复杂度

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

49 Yaroslav and Algorithm

49.1 试题来源

Codeforces 301C

49.2 题目大意

有这样一种算法，以字符串a作为输入，由一些命令组成。

第i个命令形如“s_i >> w_i”或”s_i <> w_i”，s_i、w_i是长度不超过7的字符串（可以为空），由数字或者字符”?”组成。

这个算法每次寻找一个最小的i，满足s_i是a的子串（找不到就终止算法），然后a中s_i第一次出现的位置会被w_i替换。如果这个命令中间是”<>”，那么算法终止。

你要给出若干命令，实现给一个整数+1的操作。

命令条数不超过50。

49.3 算法分析

源代码

```
#include <cstdio>
#include <algorithm>

using namespace std;

int main() {
    for (int x = 0; x <= 8; ++x)
        printf("%d??<>%d\n", x, x + 1);
    printf("9??>>??0\n");
    printf("??<>1\n");

    for (int x = 0; x <= 9; ++x)
        printf("?%d>>%d?\n", x, x);
    printf(">>?\n");
}

```

如图，上述代码即可打印出所求的命令。

基本思想是，现在最左端生成一个指针”?”，然后用命令把”?”遍历到最右端，给最右端的数+1，然后指针换为”??”，接着向左一直进位，直到找到第一个非9的数字，终止算法。

49.4 时空复杂度

时间复杂度：$O(1)$

空间复杂度：$O(1)$

50 Tennis Rackets

50.1 试题来源

Codeforces 309D

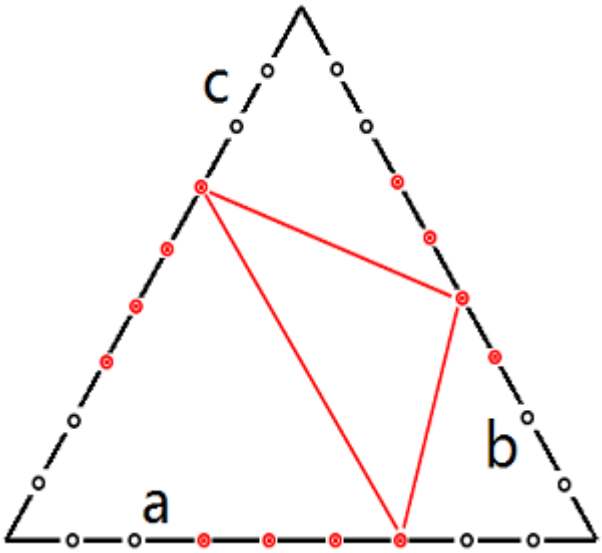
50.2 题目大意

一个等边三角形，每条边上有n ($n \leq 32000$)个均匀分布的点，靠近每侧顶点的前m ($m \leq \lfloor \frac{n}{2} \rfloor$) 个点为坏点，其他点为好点。

现在要从每条边上选一个好点，使得三个点连成一个钝角三角形，求合法方案数。

50.3 算法分析

注意到n并不是特别大，又难以找到复杂度较优的算法，可以考虑小常数的平方级算法。



如图所示，由于钝角三角形有且仅有一个钝角，不妨设钝角在底边，我们枚举a，再从小到大枚举b，由于要保持底角是钝角，c是单调递增的。那么，为了维护c的值，我们需要一个常数较小的判定钝角的方法。一个好办法是利用余弦定理推出底角cos值的表达式，然后通过cos值的正负来判定。

还有一个减少常数的优化：枚举a时只需枚举到n的一半。因为另一半的情况是对称的。

50.4 时空复杂度

时间复杂度：$O(n^2)$

空间复杂度：$O(n)$

51 Yaroslav and Arrangements

51.1 试题来源

Codeforces 301E

51.2 题目大意

定义一个数列$a_{1\sim r}$是$good$的，当且仅当它满足：

- $|a_1 - a_2| = |a_2 - a_3| = ... = |a_r - a_1| = 1$
- $a_1 = \min_{i=1}^{r} a_i$

现在求有多少个数列$b_{1\sim l}$，满足：

- $1 \le l \le n$
- $1 \le b_i \le m$
- $b_i \ge b_{i-1} \quad (i > 1)$
- b通过重排列后能形成至少一个、至多k个$good$数列。

51.3 算法分析

由于b非降，我们直接考虑数字$1 \sim m$各自选几个。

我们发现最棘手的是方案数不多于k这个限制，仔细观察一下，一个$good$序列对应于在数轴上从某一点出发每次往相邻的点跳最终回到起点的一条路径，并且要求这条路径的起点是最小值。

如果只需要统计这样的遍历方案数，我们可以用动态规划来解决，即定义$f_{i,j}$表示只考虑数字$1\ i$，最后一个数字还需要遍历j次，的方案数，转移的时候我们把$i+1$给塞到路径中去。

令$dp_{i,sum,rest,way}$表示：考虑完数字$1 \sim i$，一共选了sum个数字，最后一个数字还需要走到$rest$次，当前遍历的方案数为way，的方案数。转移的时候，枚举下一个数字选x个，根据可重组合的知识，能转移到状态$dp_{i+1,sum+x,x-rest,way*\binom{x-1}{x-rest}}$。

注意到为了保证第四维不超过k，枚举的转移会很少，即便状态的$rest = 1$，对way这一维考虑均摊复杂度也是$O(\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + ...)$这样的log 级别的。

51.4 时空复杂度

时间复杂度：$O(n^2mk)$

空间复杂度：$O(n^2k)$

52 Context Advertising

52.1 试题来源

Codeforces 309B

52.2 题目大意

有一篇文章有n个单词，单词之间用空格隔开。你要从中选择连续的一段单词，写到一个r行c列的格子里，要求单词不能拆行，求最多能写多少个单词。

$n, r * c \leq 10^6$，字符总数不超过$5 * 10^6$。

52.3 算法分析

我们想一下，假如枚举了选择的第一个单词i，那么接下来肯定是能写就写，写不下就换行了。

所以，对于某个i，存在唯一的j使得从i写到j刚好要换行（j作为新的一行的第一个单词）。不妨称$j = fa_i$。

于是形成了一棵树，等价于找一个点往父亲跳$r-1$步使得边权和最大。

建树可以枚举i，单调维护j。

建树之后建立倍增表$fa_{i,j}$表示i往上跳2^j步是谁、$fw_{i,j}$表示i往上跳2^j步的边权和。

最后枚举i，就可以$O(\log n)$查询第一个单词写i此时能写多少个单词了。取最多的作为答案。

52.4 时空复杂度

时间复杂度：$O(n \log n)$

空间复杂度：$O(n \log n)$

53 Escaping on Beaveractor

53.1 试题来源

Codeforces 331D3

53.2 题目大意

不能忍受你厌恶的！小美打算逃离校园。校园是一个$b \times b$的正方形在二维平面上。每个点$x, y(0 \le x, y \le b)$属于校园。为了让路径快和有趣，小美建造了一个Beaveractor，一种有效舒适的移动方式。

校园遵守交通规则：这里有n个箭头，平行于坐标轴。每对箭头是不会相交也不会互相接触。当小美到了某个箭头，小美的方向会变成箭头的方向并移动到下一个箭头或者走出了校园。小美每个时间单位走一个路程单位，可以认为没有障碍的。

学校科学家想研究Beaveractor，他们有q个计划，每个计划为：

假设0时刻小美在(x_i, y_i)，移动向量为w_i，计算时刻t_i的时候小美的位置。

你的任务就是完成这q个计划。

$n, q \le 10^5$

53.3 算法分析

我们计算每个箭头、每个询问所射到的第一个箭头或是边界，然后得到若干关键点之后，按移动方式在关键点之间连边，得到一个由关键点组成的图。

如果得到了这张图，就只需要构建倍增表，然后每次查询的时候倍增地走就好了。

为了得到这张图，我们需要用扫描线。具体来说，比如要求出向下的箭头碰到的第一个障碍，就从下往上扫描，对于每个横向的障碍，在线段树上区间染色，然后碰到向下箭头就在对应位置询问颜色即可。

53.4 时空复杂度

时间复杂度：$O(n \log n + q \log t)$

空间复杂度：$O(n \log t)$

54 Google Code Jam

54.1 试题来源

Codeforces 277D

54.2 题目大意

你在打GCJ，有t的时间，一共有n道题，每道题有若干属性：

- S_i, ST_i - 小数据的分数、做出来所需时间
- L_i, LT_i - 大数据的分数、做出来所需时间
- F_i - 大数据写挂的概率

GCJ的罚时指的是最后一次提交正确解的时间。
每道题必须先做小数据再做大数据。
你需要求出最大期望得分，和对应的尽可能小的期望罚时。
$n \leq 1000$，$m \leq 1560$，$\leq 10^9$。

54.3 算法分析

先考虑一个基本的问题：如果确定了做哪些题，如何确定做题顺序最小化期望罚时？（因为此时期望得分是一定的）

考虑做题时间相邻的两道题i, j，令p_i表示i做对的概率，考虑这两题产生的罚时：

$$(p_j + (1 - p_j) * p_i) * t_i + p_j * t_j$$

$$= p_j t_i + p_i t_i - p_i p_j t_i + p_j t_j$$

为了满足交换i、j不会更优，有：

$$p_j t_i - p_i p_j t_i \leq p_i t_j - p_i p_j t_j$$

$$p_j t_i (1 - p_i) \leq p_i t_j (1 - p_j)$$

$$\frac{t_i(1 - p_i)}{p_i} \leq \frac{t_j(1 - p_j)}{p_j}$$

也就是按这个值排序，再一路做过去。特别地，$p = 1$的题目（小数据）会最早做。

那么就可以直接背包了，按上述的值排序，$f_{i,j}$表示前i题花了j的时间的最大期望得分，同时存下此时的最小期望罚时。由于已经按那个值排序了，那么新的题的小数据会加在做题序列的最前面，大数据会加在做题序列的最后面，也就可以维护了。

54.4 时空复杂度

时间复杂度：$O(nt)$

空间复杂度：$O(n)$

55 Candies Game

55.1 试题来源

Codeforces 341E

55.2 题目大意

Iahub正在玩一个不寻常的游戏。一开始他有n个编号从1到n的箱子，每个箱子有一定数量的糖果在里面。用一个数组a来描述，a_i表示箱子i里面的糖果数目。

这个游戏的目的是将所有的糖果放入恰好两个箱子里。剩下的$n-2$个箱子里面必须一个糖果都没有。Iahub可以执行一系列操作或不执行任何操作。每一次他可以选择两个不用的箱子ij。不妨假设$a_i \le a_j$。然后Iahub从箱子j拿a_i颗糖果放入箱子i中。显然，当两个箱子的糖果数目相同时，箱子j便空了。

你的任务是给出一系列操作使得Iahub能达到他的目的。如果Iahub无法获得达到目的，输出-1.请注意假如有解，你并不需要输出操作最小的方案。

$n \le 1000$，$\sum_{i=1}^{n} a_i \le 10^6$。

55.3 算法分析

考虑任意3个非0的箱子，设里面的糖果为a,b,c（$a \le b \le c$）。

令$t = \lfloor \frac{b}{a} \rfloor$，然后从低到高枚举$t$的每一二进制位，对应二进制是1就用$a$拿$b$，否则用$a$ 拿c，这样得到了一个数$b \mod a$，也就是意味着最小的数变得更小了。

所以这样搞至多$O(\log)$次就能得到一个0。

至此我们能把3个正数变成2个，问题就已经解决了。

55.4 时空复杂度

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

56 Tournament-graph

56.1 试题来源

Codeforces 323B

56.2 题目大意

你要构造一个有n个结点的竞赛图，使得对任意两个结点u和v（$u \neq v$），从u到v的最短距离不超过2。

竞赛图就是基图为无向完全图的有向图（每对结点之间有一条有向边相连，且无自环）。

$n \leq 1000$

56.3 算法分析

$n \leq 4$的情况可以手工算出。

下面我们把点从0开始标号。

如果n是奇数，那么i向$(i+j) \mod n$ $(1 \leq j \leq \lfloor \frac{n}{2} \rfloor)$连边。

否则，先构出$n-1$的答案，然后考虑新加的一个点，和$0 \sim n-2$分别连边，奇数进偶数出。

容易发现这样是符合条件的。

56.4 时空复杂度

时间复杂度：$O(n^2)$

空间复杂度：$O(n^2)$

57 Piglet's Birthday

57.1 试题来源

Codeforces 152E

57.2 题目大意

Piglet要过生日了，Winnie想送给他一些蜜罐。

Winnie的家里有n个架子，每个架子上都有一些（或没有）蜜罐。初始时所有蜜罐都是装满蜜的。Winnie一共去了q次架子；第i次Winnie会先去第u_i个架子，拿走k_i个蜜罐，把这些蜜罐中的蜜吃掉，然后把这些蜜罐都放到第v_i个架子上。当Winnie拿走蜜罐时，他会在第u_i个架子上所有k_i个蜜罐的集合中等概率选择一个集合，然后把集合中的k_i个蜜罐拿走。

Winnie想知道，每次操作后，架子上所有蜜罐都被吃完的架子的期望个数是多少。

$n, q \le 10^5$，$a_i \le 100$，$k_i \le 5$。

57.3 算法分析

$f_{i,j}$表示架子i上还有j罐满的的概率。

首先随机拿k罐可以看作拿了k次每次随机拿，所以我们模拟拿k次就好了，每次拿维护一下f，即分为拿到了满的还是空的进行讨论并转移。

57.4 时空复杂度

时间复杂度：$O(qa_ik_i)$

空间复杂度：$O(na_i)$

58 Monsters and Diamonds

58.1 试题来源

Codeforces 325C

58.2 题目大意

Piegirl 发现一只怪物和一本关于怪物和馅饼的书。当她在读这本书的时候，她发现有n种怪物，每种都有一个唯一的1到n的编号。如果你喂怪物一块饼，它就会分成一定数量的怪物（可能为零），以及至少一个多彩的钻石。怪物们可能存在多种分裂方式。

最开始Piegirl 有且只有一只怪物。她先喂了一块饼，它随之分裂。对于分出来的怪物，继续喂饼，直到它们都分裂为钻石。然后她把所有钻石收集起来。

你将得到m条规则描述不同怪物的分裂方式，每种怪物至少有一种分裂方式。分裂的时候，如果有多种方式，Piegirl 可以选择任意一种。

你的任务是：对于每种怪物，确定以其为起始，Piegirl 可以得到的钻石最少最多分别是多少。Piegirl 有无限多的饼。

（如果有数字大于314000000 （但不是无穷大），则用314000000 代替。）

$n, m \leq 10^5$

58.3 算法分析

怪物分裂的时候显然是产生了若干个递归子问题。

先考虑求最小值f_i，由于这是个拓扑序混乱的DP，但转移是满足值序的，所以直接跑dijkstra 算法求解。

具体来说，就是先把只产生钻石的怪物加入堆，然后每次取出一个f 值最小的怪物z，将其标记为已处理完，然后类似拓扑排序的方法，给所有能转移到z的怪物出度减一，如果此时它没有出度了说明可以计算f 值了，于是计算后塞进堆。

有了最小值，我们就知道每个点出发是否可以不inf了。最大值直接记忆搜即可，如果搜到了正在处理的点说明可以inf，否则一路转移即可。

58.4 时空复杂度

时间复杂度：$O(m \log n)$

空间复杂度：$O(n+m)$

59 BerDonalds

59.1 试题来源

Codeforces 266D

59.2 题目大意

一张n个点m条边带权的无向图，求一个点（可以在边上），使得它到最远的点最近。

你只需要输出这个距离。

$n \leq 200$

59.3 算法分析

先用Floyd算法求出两两点对的最短路长度。

枚举一条边(u,v)，这时候每个点都有两个属性du_i, dv_i表示到两端分别的距离。

由于是要让最远点最近，假设我们二分答案K，那么所有点被分成了两种：

1. $du_i \leq K$
2. $du_i > K$

对于第一类点，我们不用理睬，对于第二类点，我们关心的是其中dv_i最大值。

至此我们就得到了算法，枚举一条边(u,v)，所有点按du_i排序，维护排序后dv_i的后缀最大值。然后我们枚举走到u的最远点，也就得到了走到v的最远点，利用这两个距离可以$O(1)$计算出这条边上所应选择的点。

注意到对于每个不同的u才需要排序一次，所以一共只需要排序n次。

59.4 时空复杂度

时间复杂度：$O(n^3)$

空间复杂度：$O(n^2)$

60 Fetch the Treasure

60.1 试题来源

Codeforces 311C

60.2 题目大意

有h间密室排成一行，有n个宝藏，第i个宝藏在第a_i个密室，价值c_i。

一开始你有一个技能k。

有m个操作，每个操作有下列3种类型：

1. 增加技能x
2. 让第x间密室的宝藏价值减少y
3. 查询当前可以到达的密室中价值最大的宝藏并拿走，如果有多个选最左的

设你的技能集合为s，你能走到的位置为$1+\sum_{i=1}^{|s|}(v_i s_i)$，$v_i \geq 0$。

操作1不超过20次。

$n,m \leq 10^5$，$k \leq 10^4$，$h \leq 10^18$

60.3 算法分析

注意k不大，考虑把所有位置都拿到模k意义下讨论。

令dis_i表示走到模k等于i的位置，至少要走多远。注意到这是一个最短路的模型，可以每次新加一个技能之后，重新跑一遍dijkstra来维护。

然后我们把当前可以够得着的物品用一个堆维护。3操作就直接从堆中取出；2操作直接减少其权值（如果已在堆中再调整堆）；1操作后枚举每个物品用dis判断是否够得着，然后重建堆。

60.4 时空复杂度

时间复杂度：$O(m \log n)$

空间复杂度：$O(n)$

61 PE lesson

61.1 试题来源

Codeforces 316D

61.2 题目大意

有n个小朋友，第i个小朋友拿着一个标号为i的球。

现在他们之间可以进行若干次抛球，即每次选定俩小朋友a,b，然后他们把手中的球抛给对方。

每个小朋友有一个体力值（1或2），表示最多能抛出去多少次球。

求能形成多少种不同的持球方式。答案模10^9+7。

$n \le 10^6$

61.3 算法分析

我们直接考虑，对于给定的持球方式，如何判断是否可行。

用置换的有关知识，我们不难发现，一个持球方式可行，当且仅当其对应的置换拆成循环的形式后，每个循环内有至多2个1。

所以我们先把体力值为1的小朋友们分组，再逐个插入体力值为2的小朋友。

令$I(n)$表示n个体力值为1的小朋友的分组方案数，考虑到新加入的小朋友要么自己一组，要么和之前某个人一组，所以有：$I(n)=I(n-1)+(n-1)*I(n-2)$。

然后插入体力值为2的小朋友们。由于这些小朋友随便插入到哪个位置都可行，还可以自己新建一个循环，所以第i个小朋友给答案乘上(cnt_1+i)。

61.4 时空复杂度

时间复杂度：$O(n)$

空间复杂度：$O(n)$

62 Two permutations

62.1 试题来源

Codeforces 323C

62.2 题目大意

你有两个排列p、q。

m次询问，每次给定l_1, r_1, l_2, r_2，问在p中位置为$l_1 \sim r_1$、q中位置为$l_2 \sim r_2$的数字的个数。

强制在线。

$n \leq 10^6$，$m \leq 200000$。

62.3 算法分析

对于每个数字，有两个属性：

1. x_i - 在p中出现的位置。
2. y_i - 在q中出现的位置。

把这俩属性看做点的坐标，那么每个询问等价于问平面内某个矩形内部点的个数。

令$Tree(i)$表示$1 \leq x \leq i$的点的y建立的权值线段树，用函数式线段树即可求出$Tree(1 \sim n)$。

那么每次询问就直接在$Tree(r_1)$和$Tree(l_1 - 1)$里查询$[l_2, r_2]$的区间和，再相减即可。

62.4 时空复杂度

时间复杂度：$O(n \log n + m \log n)$

空间复杂度：$O(n \log n)$

63 Pumping Stations

63.1 试题来源

Codeforces 343E

63.2 题目大意

给一张无向图，n个点m条边，边带权。

你需要找到一组$v_{1\sim n}$，最大化：

$$\sum_{i=1}^{n-1} maxflow(v_i, v_{i+1})$$

$n \le 200$，$m \le 1000$。

63.3 算法分析

首先，最大流等于最小割。考虑全局最小割，假设分成的两个集合为S、T，那么$mincut(u \in S, v \in T)$ 就都得到了，每个集合再递归求解。这样一来，我们得到了一个树形的结构，每两个点在原图中的最小割等于树上路径的min。这个树就是Gomory-Hu树。

进一步地我们可以发现，任意两个最小割是不会互相跨立的。即假设(U, V)和(A, B)是两个最小割，那么$U \cap A$、$U \cap B$、$V \cap A$、$V \cap B$不会全非空。这个可以画个图用不等式证明。

于是我们得到一个$O(n*maxFlow(n))$的算法来对于每一对i, j求出$mincut(i, j)$。即每次从当前集合中随便选两个点当做源和汇，然后求出最小割，设两部分分别为S、T，用这个最小割更新$mincut(u \in S, v \in T)$，然后S、T分别递归做下去。

得到了每个$mincut(i, j)$也就能方便地建出Gomory-Hu树了。最后，每次从树中选出最小边(u, v)，用调整法可以证明这条边只会被经过一次，所以先对子树u递归构造，然后经过这条边(u, v)，再对子树v递归构造。

63.4 时空复杂度

时间复杂度：$O(n * maxFlow(n))$

空间复杂度：$O(n^2)$

64 Berland Traffic

64.1 试题来源

Codeforces 267C

64.2 题目大意

现有一个n个点m条边的网络。每条边有一个容量限制，对于一条边(x,y)，设其流量为t，容量为c，如果t大于0则是从x流向y，反之就是从y流向x，但t的绝对值不能超过c且可以不是整数。

网络中节点1是源点，节点n是汇点，对于除$1,n$外的所有点，流入的流量等于流出的流量。这个网络有一个奇怪的性质，对于任意一对节点(x,y)，x到y的路径上流量之和不会随着选择不同的路径而改变（有可能有小于0的流量，流量的符号取决于x到y路径上这条边的方向）。

求满足上述条件且流过该网络的流量尽可能大的方案以及最大的流量是多少。

$n \le 100$，$m \le 5000$

64.3 算法分析

抛开容量不看，我们注意流量的特殊限制，说明每个点有一个类似势能的东西，不妨设为x_i。

我们强制令源的势能为0，汇的势能为1，对于每个其他的点，用流量平衡可以得到一个关于x的方程。把方程组高斯消元解出来就得到了所有点的势能。

最后，每条边有一个算出来的流量f，为了不超过c，我们要记录最大的f/c，给所有势能除以这个值。

得到了势能就自然得到了流量了。

64.4 时空复杂度

时间复杂度：$O(n^3)$

空间复杂度：$O(n^2)$

65 Road Repairs

65.1 试题来源

Codeforces 240E

65.2 题目大意

有一个国家有n个城市和m条单向道路，首都是1。

每条道路可能修好了也可能是坏的。

你需要求出至少得再修好多少条坏的道路，使得1号点出发能到达所有点。

$n \leq 10^5$

65.3 算法分析

一眼就是一个最小树形图的模型。

众所周知最小树形图是可以$O(nm)$的。

事实上还有个用左偏树优化的$O(m \log n)$的。

65.4 时空复杂度

时间复杂度：$O(m \log n)$

空间复杂度：$O(m)$

66 Evil

66.1 试题来源

Codeforces 329E

66.2 题目大意

平面上n个点，第i个点坐标(x_i, y_i)。

两点的距离定义为曼哈顿距离。

求一条最长的哈密尔顿回路。

$n \le 10^5$，$0 \le x_i, y_i \le 10^9$

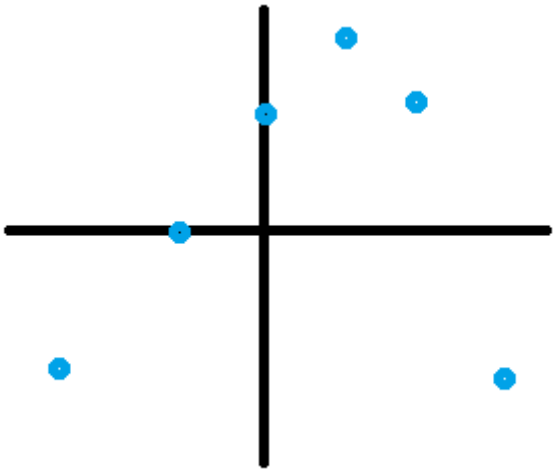
66.3 算法分析

先考虑一个弱化版，一维的情况怎么做。

注意$|x_1 - x_2|$这种式子，x_1, x_2的贡献必然是一正一负。由于是哈密尔顿回路，每个值必然会出现两次。所以我们把每个x写两遍，然后排好序，排成一排，前一半加上负号，后一半加上正号，这样求出来的和必然是最优的情况了。有没有办法做到呢？定义比中位数大的为黑，其余为白，那么访问的时候一黑一白就好了。

接下来考虑二维的情况。

为了避免烦人的讨论，不妨想象给所有点坐标进行微小的抖动，这样在不影响答案的前提下使得没有相同的坐标了。



二维的目标是，尽量使x、y都满足一维的最优情况。

我们求出x中位数、y中位数，把平面分成了4个区域。那么符合要求的移动就是$A-C$或$B-D$。

对n的奇偶性、以及点的边界情况进行讨论，最终会发现要么能取得最优情况，要么能取得次优情况。

66.4 时空复杂度

时间复杂度：$O(n\log n)$

空间复杂度：$O(n\log n)$

67 Random Ranking

67.1 试题来源

Codeforces 303E

67.2 题目大意

有n个人来打比赛，第i个人的得分在$[L_i, R_i]$里等概率随机。

分数越高的排名越靠后。

求一个$n \times n$的矩阵，第i行第j个数表示第i个人拿到第j名的概率$prob_{i,j}$。

$n \leq 80$，$0 \leq L_i < R_i \leq 10^9$

67.3 算法分析

我们把所有的端点离散化出来，枚举一个人，再枚举这个人的得分处于哪一段。

接下来其他所有人，要么一定排名在这个人之前，要么排名一定在这个人之后，要么会有3个概率pl_i, pm_i, pr_i分别表示分数小于这一段的、分数处于这一段、分数大于这一段的概率。

然后考虑$dp_{i,j,k}$表示考虑完前i个人、有j个人和自己分数同段、有k个人分数比自己小的概率，最后把$dp_{n,j,k}/(j+1)$转移给$prob_{me,k\sim k+j}$。

考虑这个dp如何求，暴力求是$O(n^3)$的，但注意到这是个类似背包的转移，可以用分治做到$O(n^2 \log n)$。

67.4 时空复杂度

时间复杂度：$O(n^4 \log n)$

空间复杂度：$O(n^2)$

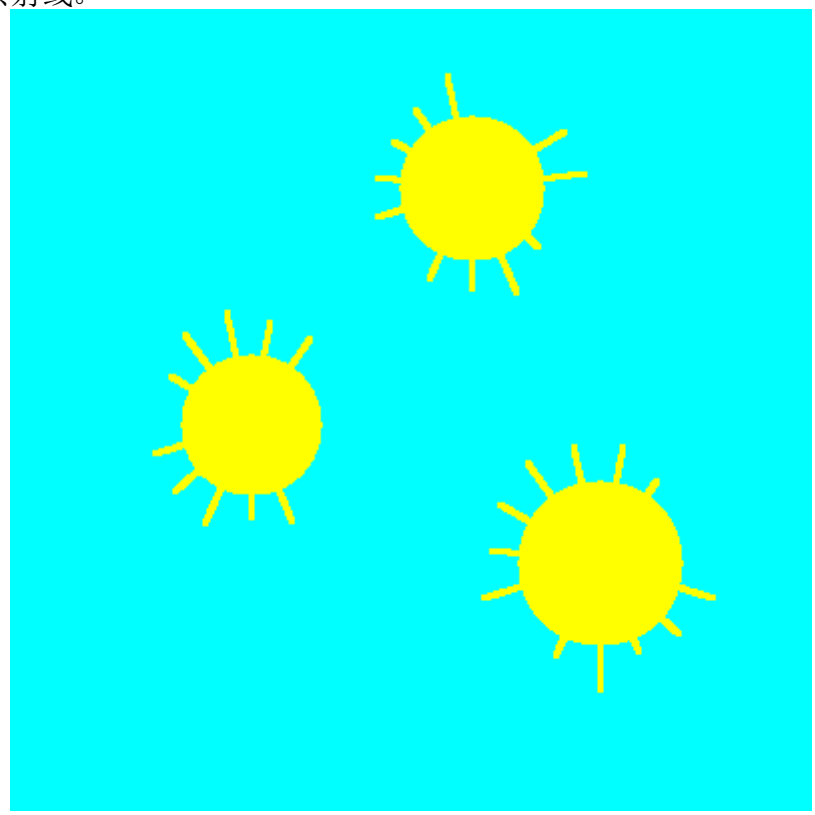
68 Suns and Rays

68.1 试题来源

Codeforces 316F3

68.2 题目大意

给你一张类似下面的图片，你需要找出有多少个太阳，以及每个太阳有多少条射线。



保证:
没有两个太阳有共同点。
射线的宽度为3像素。
太阳的轴的长度将介于40和200像素。
没有两条射线相交。
所有射线的长度将在10和30像素。

68.3 算法分析

定义一个“侵蚀”操作，表示将所有和蓝色区域有接触的黄色像素给抹去。

注意到侵蚀若干次后，图像上就只剩下了若干个圆。这时候数一数连通块个数就得到太阳个数了。

假设我们把侵蚀后的太阳涂黑，再和侵蚀前的图像对比，我们发现，如果把黑太阳扩大一点，原来的每条射线就都成了一条独立的连通块。所以枚举每个太阳，然后dfs出连通块个数就是射线个数了。

68.4 时空复杂度

时间复杂度：$O(XY)$

空间复杂度：$O(XY)$

69 Levko and Game

69.1 试题来源

Codeforces 360E

69.2 题目大意

一张n个点m条边的带权无向图，有k条边的权值可以改变，其中的第i条可以在$[L_i, R_i]$之间选取。

俩人在玩游戏，1站在s_1，2站在s_2，他们要比比谁能更先到达f。

现在你是1，你可以自由改k条边的权值。问能否胜利，如果不能的话能否平局，还是说必败。

$n, m \leq 10^4$

69.3 算法分析

先讨论能否胜利。

一开始由于每条道路不知道自己会不会利用到，不妨全设成R_i。

定义一个点被占据，当且仅当$d1_i < d2_i$（$d1_i$表示$dist(s1, i)$，$d2$同理）。于是每次看一看k条道路哪些的起点已经被占据，然后把那些起点被占据的道路的长度缩小为L_i，从而进一步扩大占据的范围。

如果占据了f，就证明目的达成了。否则，如果无法再扩张下去，说明目的无法达成。

如果不能胜利，再把目的降为平局，类似地再做一遍（只需要把<改成≤即可）。

上述过程如果直接暴力实现的话，最坏要做k次最短路算法。

注意到扩展和最短路是可以同时进行的，所以就只要跑一次最短路了。

69.4 时空复杂度

时间复杂度：$O(m \log n)$

空间复杂度：$O(n + m)$

70 Optimize!

70.1 试题来源

Codeforces 338E

70.2 题目大意

```
getAnswer(a[1..n], b[1..len], h)
  answer = 0
  for i = 1 to n-len+1
    answer = answer + f(a[i..i+len-1], b, h, 1)
  return answer

f(s[1..len], b[1..len], h, index)
  if index = len+1 then
    return 1
  for i = 1 to len
    if s[index] + b[i] >= h
      mem = b[i]
      b[i] = 0
      res = f(s, b, h, index + 1)
      b[i] = mem
      if res > 0
        return 1
  return 0
```

请帮助优化这段代码。

$1 \le len \le n \le 150000$，$1 \le a_i, b_i, h \le 10^9$。

70.3 算法分析

观察一下这段代码，大概就是说，两个数字加起来如果不小于h就能匹配，然后求有多少个i满足$a_{i\sim i+len-1}$和b能形成完备匹配。

为了简化问题，先给b从大到小排序，然后求出p_i表示a_i选的b_j中j最大只能是多少。

直接上Hall定理，对于有一段p，只要对于每个x满足$\sum_{i=1}^{len}[p_i \le x] \le x$就好了。想象一个数组$z_i$表示$p_j = i$的$j$的个数，那么我们实际上就是要维护$i - z_i$的最小值（非负才存在完备匹配）。

所以直接扫描p数组，每次尾部加一个，头部删一个，然后线段树维护一下$i - z_i$这个序列的区间最小值就好了。

70.4 时空复杂度

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

71 Positions in Permutations

71.1 试题来源

Codeforces 285E

71.2 题目大意

P是n个互不相同且不超过n的正整数的一个排列。我们设排列P的第i个元素为P_i，n为排列的长度。

我们称排列中的第i个位置是完美的，当且仅当$|P_i - i| = 1$。

请求出长度为n的而且完美的位置数刚好为k的排列数是多少。答案要求取模$10^9 + 7$。

$n \leq 1000$

71.3 算法分析

有这样一种生成排列的方式：假设当前有一个$1 \sim n-1$的排列，在末尾放一个n，然后和前面某个位置上的数交换一下。

这种生成方式是很容易搞出DP的。

基于上面的思想，一个很基本的思路是，$f_{i,j}$表示生成完了$1 \sim i$的排列、有j个位置是完美的方案数。但是我们发现还缺了一些信息。

首先，设我们新插入的数为n，对于n来说，n、$n-1$、其他，是三种不同的情况。前两种会形成一个新的完美位置，第三种不会。那么注意$n-1$，如果那个位置换过来的数字恰好是$n-1$，完美位置数还会+1。所以状态里还要记录一个“位置i上的数是否恰好为i”。

还有一种特殊的交换，就是n特意去找数字$n-1$所在的位置换，这样有可能拆散之前的完美位置。所以我们还得记录“数字i所在位置是否是完美位置”。

最后，如果n交换到位置$n-1$，换来的是一个普通的数，为了确认是否少了完美位置，还得记录“位置i是否是完美位置”。

我们发现记录这三个布尔变量就能轻松地转移了。

71.4 时空复杂度

时间复杂度：$O(n^2)$

空间复杂度：$O(n^2)$

72 Tape Programming

72.1 试题来源

Codeforces 238D

72.2 题目大意

有一个编程语言，程序由数字和“¡”、“¿”组成。最初有一个指针在串的最左端，方向向右。

重复以下操作直到指针指向串外：

1. 如果当前位置是数字，输出这个数字，然后指针沿着原方向移动，该数字减1。如果该数字为原来为0则删除该数字；
2. 如果指针位置是“¡”、“¿”，那么指针方向改为对应方向并继续移动。如果新位置也不是数字，则删除当前位置的字符。

给一个串$s_{1\sim n}$，q组询问，每次询问运行s_{l_i,r_i}后每个数字会被输出多少次。

72.3 算法分析

我们考虑直接对原串跑一遍，注意一个询问l,r，当指针第一次走进l的时候等价于开始了这个子程序，当指针第一次走到$l-1$或者$r+1$的时候等价于跑完了这个子程序。

所以我们假设原串的左边补了足够多的“¿”，然后在原串上模拟，每个位置开一个*vector*记录到达该位置的时间点们。

查询的时候直接找到第一次进入l时间$t0$，再在两端位置的*vector*里分别二分到时间$t0$后第一次到达，于是就求出了所询问子程序终止的时间。在原串模拟时顺便记录每种数字出现的前缀和，相减就能求出区间和了，即该子程序的答案。

72.4 时空复杂度

时间复杂度：$O(n+m\log n)$

空间复杂度：$O(n+m\log n)$

73 The Great Julya Calendar

73.1 试题来源

Codeforces 331C

73.2 题目大意

给一个数字n，每次你可以给n减去n的十进制表示下的某一位上的数，求最少的操作数把n变成0。

$n \le 10^18$

73.3 算法分析

首先容易得到一个贪心的想法，每次减掉当前数十进制表示中最大的一位。令f_x表示x所需的最少操作次数，那么我们只要证明f单调非降即可。

假设$f_{1\sim y-1}$都满足单调非降了，考虑f_y和f_{y-1}，如果y和$y-1$只有个位不同，那么操作一次后显然$new(y) \ge new(y-1)$。否则，说明$y-1$的个位是9，y个位是0，$new(y-1) = y-1-9$，$new(y) \ge y-9$，所以$new(y) \ge new(y-1)$。归纳出来就证明了f单调非降。

接下来想下怎么优化这个过程，脑补一个状态$f_{p,x}$表示之前有一个数p可以候选，求把x减到≤ 0的步数和对应的最终结果。一开始要求的是$f_{0,n}$，然后转移到$f_{n_1,n_{2\sim|n|}}$，当后面被减到小于0的时候就要退位了，n_1会减少1，设返回的最终结果为x，然后继续调用$f_{n_1,10^{|n|-1}-x}$。

转移是$O(1)$的，考虑状态量，这样记忆搜的话每个x要么是n的一个原封不动的后缀，要么形如99999x，一共是$O(\log n)$种可能。

73.4 时空复杂度

时间复杂度：$O(\log n)$

空间复杂度：$O(\log n)$

74 Polo the Penguin and Lucky Numbers

74.1 试题来源

Codeforces 288E

74.2 题目大意

定义幸运数字是正整数,同时它们的十进制数表示只能包含幸运数字4和7。例如,数字47, 744, 4是幸运数字,而5, 17, 467不是。

设l到r间幸运数字为$a_{1\sim n}$，求$a_1 * a_2 + a_2 * a_3 + ... + a_{n-1} * a_n$。答案模$10^9 + 7$。

$1 \le l \le r \le 10^{100000}$

74.3 算法分析

区间询问先转为两次前缀询问的差。

设$f_i = a_i * a_{i+1}$，那么所求贡献即$\sum f_i\ (i < l)$，按第一次$< l$的位置分类统计，即枚举一个公共前缀pre。那么当前这一位l必须是7，填必须填4，之后的就4、7任取了。于是推推式子就可以统计了。

74.4 时空复杂度

时间复杂度：$O(\log l + \log r)$

空间复杂度：$O(\log l + \log r)$

75 Race

75.1 试题来源

Codeforces 241F

75.2 题目大意

TOC是一个长方形的城市，有着$n*m$个方格状的街区。

若干建筑物，双向直线的道路以及交叉路口组成了TOC。

每个建筑物：占恰好的一个街区。

每条道路：所有道路的宽度都为1个街区，并且道路都是水平的或竖直的。

每个交叉路口：占一个街区，位于道路的交汇处。

我们称两个街区是相邻的，当且仅当它们之间存在着公共边。

没有两条道路或两个交叉路口是相邻的。

在每年的狂欢节中，TOP总会按照一条特殊的路线游行。

TOP会从一个街区出发，接下来经过若干个交叉路口，最终在一个街区停下。

TOP知道从某一个街区到其相邻的街区所花费的时间。

同时，从交叉路口需要花费1分钟来到达相邻的街区。

TOP不能经过有建筑物的地方。

我们知道TOP的初始与结束位置，也知道TOP途中经过的交叉路口的顺序。

但他完成游行之后，他会一直呆在结束位置。

你的问题是，找出在他出发了k分钟之后，位于哪个位置。

TOP总会沿着最短的道路经过给定的交叉路口，而到达终点。

注意到TOP可能会访问某些格子多于一次。

75.3 算法分析

直接模拟就行了。

先记录每个字母的坐标x_i, y_i。

然后按照输入的字符串，一分钟一分钟地朝着对应方向走就可以了。

75.4 时空复杂度

时间复杂度：$O(nm+k)$

空间复杂度：$O(nm)$

76 Cubes

76.1 试题来源

Codeforces 243D

76.2 题目大意

平面直角坐标系内画着一个以 $(0,0)$ 和 (n,n) 为对角线的大正方形，大正方形被均分成了 $n \times n$ 个单位正方形。每个单位正方形上竖着叠了若干个单位正方体。

现在有一组平行光从无穷远处以向量 $(vx,vy,0)$ 射过来，问照亮了多少个单位正方体。

$n \leq 1000$

76.3 算法分析

对于每一叠正方体，可以简化成3个量：$[l_i,r_i]$表示投影在光线方向的区间，h_i表示竖直的高度。投影可以用点积求。

那么我们按光线射入方向从近到远加入每一叠正方体，每次新增的可见正方体数量为$h_i - h_{min}$，h_{min}表示区间$[l_i,r_i]$中最低的位置。统计完后给区间$[l_i,r_i]$全部max上一个数h_i。

区间max一个数，询问区间min，用打标记的线段树就能轻松解决了。

76.4 时空复杂度

时间复杂度：$O(n^2 \log n)$

空间复杂度：$O(n^2)$

77 Close Vertices

77.1 试题来源

Codeforces 293E

77.2 题目大意

你得到了一棵包含n个点的树，树上的每条边有一个非负边权，树上两点间路径的长度是该路径包含的边数，树上两点间路径的权重是指该路径包含的边的边权之和。

我们说两点是“相邻”的，当且仅当，存在一条连接该两点的路径，满足该路径的长度小于等于L,且权重小于等于W。

统计有多少个点对(u,v)，满足$u<v$，且u,v是相邻的。

77.3 算法分析

我们树分治，设当前分治的点为d，统计所有经过d的点对的答案。

以d为根，每个点有两个属性x_i,y_i，分别表示到根的长度和权重，我们要统计的就是$x_i+x_j\le L$ and $y_i+y_j\le W$ and $\mathrm{LCA}(i,j)=d$的点对(i,j)个数，忽略最后那个条件，按y给所有点排序，逐个扫描，枚举i，每次加入新的可行的j，然后用树状数组维护x_j们。最后，由于统计了d的同一子树多余的贡献，再对每个子树自己做一次，得到子树内的贡献，减去即可。

77.4 时空复杂度

时间复杂度：$O(n\log n)$

空间复杂度：$O(n\log n)$

78 Pilgrims

78.1 试题来源

Codeforces 348E

78.2 题目大意

给一棵树，n个点，有m个黑点。

你可以摧毁一个白点。

如果一个黑点失去了自己所有最远黑点，它就会不高兴。

求你最多能让多少个黑点不高兴，并求出对应的方案数。

$n \leq 10^5$

78.3 算法分析

首先我们可以DP求出$down_v$表示v子树内最深的黑点，up_v表示v子树外最远的黑点，进一步可以顺便求出$downP_v$表示v子树最深黑点们的LCA，upP_v表示v子树外最远的黑点们的LCA（把v看做根）。

考虑一个黑点v，摧毁哪些点会让它不高兴。

如果v子树内和子树外都有最远黑点，那么怎么摧毁都无法让它失去最远点。

否则，设那一群最远黑点的LCA是p，则链$v-p$上每个点被摧毁都将导致v失去最远黑点。

所以DP完之后枚举每个点v，要进行一次树上链加值。用点事件标记即可。

78.4 时空复杂度

时间复杂度：$O(n)$

空间复杂度：$O(n)$

79 Flights

79.1 试题来源

Codeforces 241E

79.2 题目大意

有一张n个点m条边的有向拓扑图，每条边边权为1。

你打算把某些边的权值改为2，使得每一条从1号点到n号点的路径长度都相等。

$n \leq 1000$，$m \leq 5000$。

79.3 算法分析

首先，我们只关注1号点能到且能到n号点的哪些点，其余点丢掉。

由于要求每条路径都相等，每个点必须有个势能d_u，表示1到u的每条路径都是这个长度。

对于每一条边$(u-v)$，可以转化为如下限制：

- $d_v \leq d_u + 2$
- $d_u \leq d_v - 1$

然后就是一个差分约束系统的模型了。即，把上述式子看成最短路中的约束不等式，那么问题有解当且仅当图中无负权环。

用Bellman-Ford判负环就可以了。

如果有解，最后每条边$(u-v)$的边权为$d_v - d_u$。

79.4 时空复杂度

时间复杂度：$O(nm)$

空间复杂度：$O(n+m)$

80 Wall Bars

80.1 试题来源

Codeforces 268D

80.2 题目大意

Manao在一家建筑公司工作。最近，有一项在儿童公园内建设Wall Bar的订单。Manao被委托设计一个能使公司最节省资金的建设方案。

在回顾了Wall Bar 的正常规格之后，Manao发现许多要求是模糊的，决定按照公司的利益处理。他的最终设计描述如下：

1. 我们定义单位长度，建筑物中间那根管子高度为n。
2. 在高度为$1, 2...n$的地方，恰好有一根水平的横杆从中间的杆子连向四个方向中的某一个预先固定好的杆子上。
3. 如果两根横杆的距离不超过h，且方向相同，那么一个孩子可以从一个一根横杆爬到另一根上。在地上的孩子，可以爬到任何一根高度在$1 \sim h$之间的横杠上。在Manao 的建筑物上，一个从地面出发的孩子至少能到达一根高度在$n-h+1, n-h+2...n$的横杠。

Manao想知道有多少种设计方案满足上述要求。这个数字可能会非常大，输出其除以10^9+9的余数即可。两个设计方案被认为是不同的，当且仅当在某个高度他们的横杠方向不同。

$h \le n \le 1000$，$h \le 30$。

80.3 算法分析

考虑按层DP，$f_{i,a,b,c,d}$表示考虑完前i层、四根竖杆的上一个横杆离第i层的距离分别为a, b, c, d的方案数。

这样复杂度略高，考虑优化

首先，a, b, c, d是可以排序的，即排序后可以看做同一状态。

其次，注意到每一层必须有一根横杆，所以a, b, c, d必有一个0，状态从而减少了一维。

80.4 时空复杂度

时间复杂度：$O(nh^3)$

空间复杂度：$O(nh^3)$

81 Rats

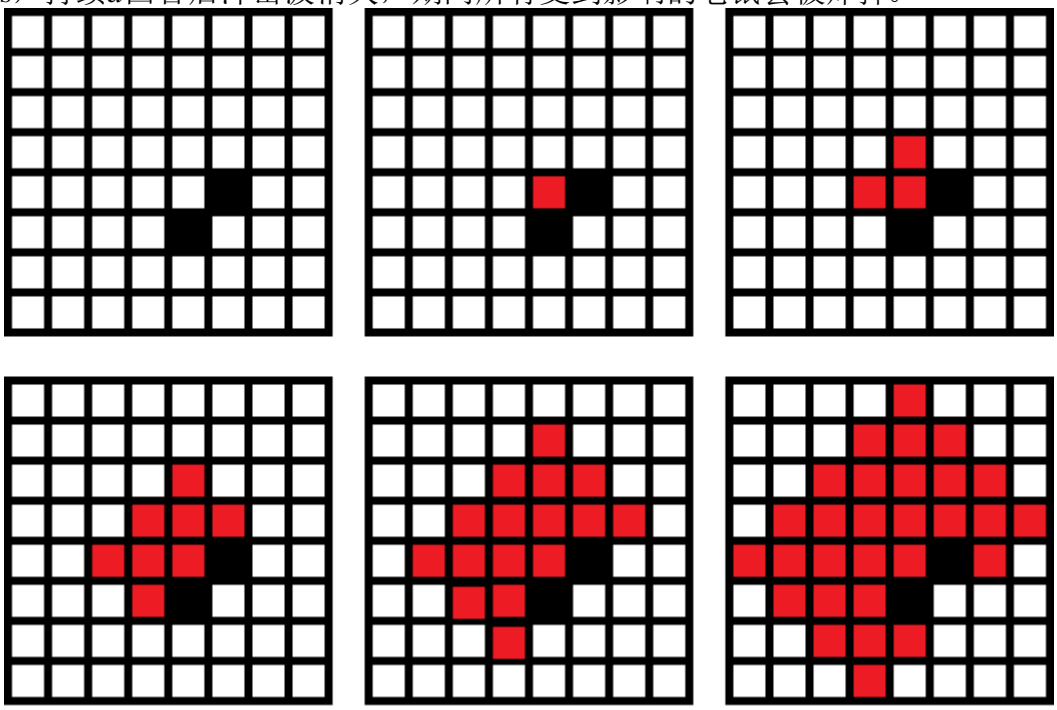
81.1 试题来源

Codeforces 254D

81.2 题目大意

有一个$n \times m$的棋盘，上面有些格子是障碍物，有些格子是老鼠。

一个炸弹投掷在(x, y)后，冲击波会从起点每回合向上下左右的非障碍格子bfs，持续d回合后冲击波消失，期间所有受到影响的老鼠会被炸掉。



你可以丢两发炸弹，问能否炸掉全部的老鼠。能还要给出方案。

$n, m \leq 1000$，$d \leq 8$。

81.3 算法分析

由于d不大，考虑一只老鼠，我们$O(d^2)$枚举炸它的炸弹在哪，然后bfs炸死一片老鼠。

用链表维护炸老鼠操作，就能$O(1)$找到任意一只剩下的老鼠。再次$O(d^2)$枚举第二颗炸弹放哪，bfs炸一片。

如果某次发现没有老鼠剩下，就输出方案。

否则，说明无解。

81.4 时空复杂度

时间复杂度：$O(nm + d^6)$

空间复杂度：$O(nm)$

82 Maximum Waterfall

82.1 试题来源

Codeforces 269D

82.2 题目大意

直角坐标系内一堵高h的足够宽的墙，有n块水平板子，第i块为$(h_i, l_i \sim r_i)$。
水可以从第i块板流向第j块板当且仅当：

1. $\max(l_i, l_j) < \min(r_i, r_j)$（二者的水平位置存在相交部分）
2. $h_j < h_i$（第j块板在第i块板下方）
3. 不存在k $(h_j < h_k < h_i)$，使得(i, k)与(k, j)均满足以上两个条件。

从i到j的流量为$\min(r_i, r_j) - \max(l_i, l_j)$，相当于二者的水平相交部分的长度。
一条路径的流量为每一步流量的最小值。
求一条从墙的顶部出发的水量最大的瀑布。
$n \le 10^5$

82.3 算法分析

容易得到一个DP，dp_i表示当前流到了第i块板子上此时最大的水量。
关键是转移的复杂度。
其实从左到右用扫描线搞一遍就能得到全部的$O(n)$的转移边。
考虑一竖直扫描线从最左端扫到最右端，那么会遇到下列事件：

1. 碰到一条线段左端，插入该线段。
2. 碰到一条线段右端，删除该线段。

对于所有线段按y维护一棵平衡树。

在插入线段时，我们先删除其前驱到后继的转移边，再添加前驱到它、它到后继的转移边。

用一个hash表维护所有转移边，最后转移一边就好了。

82.4 时空复杂度

时间复杂度：$O(n \log n)$

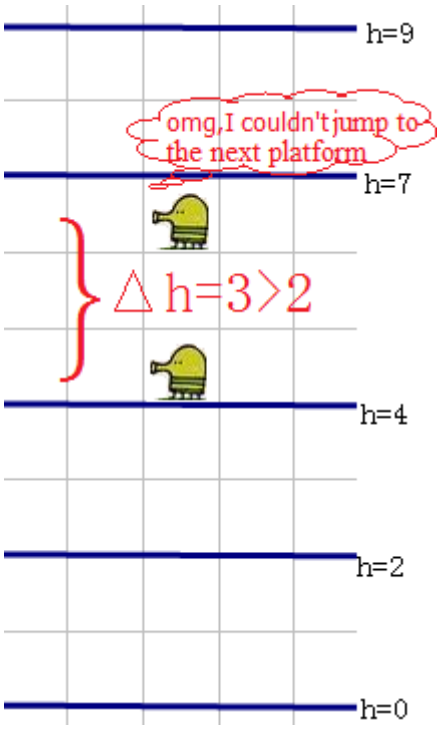
空间复杂度：$O(n)$

83 Doodle Jump

83.1 试题来源

Codeforces 346E

83.2 题目大意



考虑$n+1$个数字，第$i\ (0 \le i \le n)$个等于$ia \mod p$，保证a,p互质。

问给这n个数字排序后相邻差是否有大于h的。

T组询问。

$T \le 10^4$，$1 \le a,n,p,h \le 10^9$。

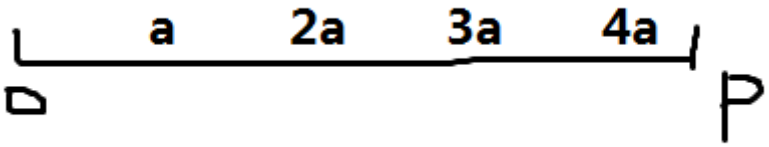
83.3 算法分析

不妨直接求出最大差是啥。

如果$a*n<p$，那最大差就是a。

否则，定义此时最大差为$solve(a,n,p)$。考虑这个怎么求。

我们把$ia \mod p$分成一轮一轮的。



每一轮的起点位移为$a \mod p$，观察并思考，我们发现最大差一定产生于图中$[3a, 4a)$这一段。

所以求出轮数cnt，直接递归下去。

```
s64 solve(s64 a, s64 n, s64 p)
{
    if (a * n < p) return max(a, p - a * n);
    s64 cnt = a * n / p;
    if (a * n % p < a * (p / a - 1))
        --cnt;
    return solve(min(a - p % a, p % a), cnt, a);
}
```

注意，这里递归下去的最大差还多考虑了最大值和模数的差。

83.4 时空复杂度

时间复杂度：$O(\log(a + p))$

空间复杂度：$O(1)$

84 Colorful Stones

84.1 试题来源

Codeforces 264D

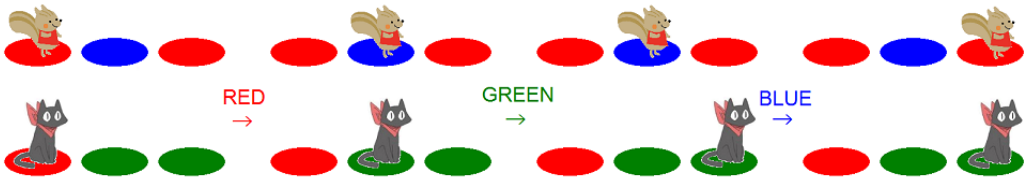
84.2 题目大意

有两串石头s、t，每个石头有三种颜色R、G和B。

两个小动物分别站在两串石头的最左端的石头上。

你每次可以指定一个颜色，然后站在指定颜色的石头上的小动物都向右前进一格。

求有多少个状态(x,y)是可达的。x,y分别表示两只小动物在自己的串上的位置。



$|s|,|t| \le 10^6$

84.3 算法分析

一个直接的想法当然是枚举x，维护可行的y了。

经过一番模拟，我们发现，任意时刻，y的可行区域都是一个区间，挖掉一些地方。

区间的r每次能走就走。

区间的l每次不得已才走。

设当前枚举的$x=i$，令字符$c_1=s_i$，$c_2=s_{i-1}$，挖掉的地方都形如c_2c_1。

所以记录一个前缀和，维护l,r，每次给答案加上区间长度，减去区间内被挖掉的数目。

84.4 时空复杂度

时间复杂度：$O(|s|+|t|)$

空间复杂度：$O(|s|+|t|)$

85 k-Maximum Subsequence Sum

85.1 试题来源

Codeforces 280D

85.2 题目大意

有一整数序列$a_{1\sim n}$。

有m个操作，每个操作有如下两种类型：

1. 把a_i赋值成val
2. 询问从区间$a_{l\sim r}$中选出至多k段不相交的连续子段的最大和。

$n, m \le 10^5$，$k \le 20$。

85.3 算法分析

用费用流建模，那么等价于流k次，每次流最大子段和$[lm, rm]$，再给$[lm, rm]$的权值乘上-1。

注意到每次选出的是最大子段和，也就是这一段的每个前缀和后缀的和都> 0，所以乘上-1后都< 0，那么之后拿最大子段和的时候，就不会出现相交的情况了。

于是每次询问我们模拟k次取最大子段和的操作。

单点赋值，区间乘-1，维护最大子段和。

这是可以用带标记的线段树解决的。

85.4 时空复杂度

时间复杂度：$O(km \log n)$

空间复杂度：$O(n)$

86 Matrix

86.1 试题来源

Codeforces 243E

86.2 题目大意

给一个$n \times n$的01矩阵。

你需要重排所有列，使得每一行的1连续。

$n \leq 500$

86.3 算法分析

把问题抽象一下，相当于有n个元素，接下来n个限制条件，每个条件要求某些个元素靠在一起。

这是一道PQ树的例题。

大致思路是：拿一棵树来维护就好了。n个东西作为叶子，然后每个dfs序对应一个排列。

然后每加入一个新限制，就暴力修改这棵树。

根据限制的集合S，设x子树的叶子集合为$L(x)$，每个节点有三种情况：

1. $L(x) \cap S = \emptyset$
2. $L(x) \cap S = L(x)$
3. 除了楼上两种

对于情况1、2，显然在子树里怎么走都没问题。

所以我们要集中精力搞第3种情况。

搞一搞会发现，会形成两类节点：

1. 儿子们可以无序访问。
2. 儿子们必须按顺序或倒序访问。

仔细讨论一下就可以每次$O(n)$维护了。

86.4 时空复杂度

时间复杂度：$O(n^2)$

空间复杂度：$O(n)$

87 The Evil Temple and the Moving Rocks

87.1 试题来源

Codeforces 329D

87.2 题目大意

有一个$n \times n$的棋盘（n为偶数），上面每个格子要么为空，要么有一个方向键。

当你激活一个方向键后，它会朝着指定方向不断移动，直到碰到边界或另一个方向键，碰到的方向键会被激活。

当一个激活的方向键运动的距离> 0再被停止的时候，会发出一声响声。

你需要构造一个局面，发出响声至少为$k^3 - k^2$，$k = n/2$。

$n \leq 300$

87.3 算法分析

把形如下面两行的构造重复$n/2$次即可：

```
>>>>_>_>_>_>_v
^_<_<_<_<_<<<<
```

激活第一行最后一个，这样每两行都能贡献很多次。

87.4 时空复杂度

时间复杂度：$O(n^2)$

空间复杂度：$O(n)$

88 Ksusha and Square

88.1 试题来源

Codeforces 293D

88.2 题目大意

有一个n个点的凸多边形，每个顶点坐标都是整数。

在凸多边形内部（包括边界）随机选两个点，求以这两个点为对角线的正方形的面积的期望。

$n \le 10^5$，坐标绝对值不超过10^6。

88.3 算法分析

正方形的面积等于对角线乘积的一半，不妨直接考虑对角线长度的平方的期望。

即$(x_1-x_2)^2+(y_1-y_2)^2$的期望。注意到x,y是互相独立的，所以可以分开算，下面只讨论x。

$$\frac{1}{n(n-1)}\sum_{i=1}^{n}\sum_{j=1}^{n|(j\neq i)}(x_i-x_j)^2$$

$$\frac{1}{n(n-1)}\sum_{i=1}^{n}\sum_{j=1}^{n|(j\neq i)}x_i^2+x_j^2-2x_ix_j$$

$$\frac{1}{n(n-1)}(2(n-1)sum_2-2\sum_{i=1}^{n}x_i(sum_1-x_i))$$

$$\frac{1}{n(n-1)}(2(n-1)sum_2-2sum_1^2+2sum_2)$$

sum_1表示所有x的和。

sum_2表示所有x的平方和。

n表示可选的x的个数。

所以我们只需要统计上述三个值。

由于坐标范围不大，考虑从左到右扫描，对于每条竖线$x = i$统计有多少个点在多边形内。由于多边形是凸的，一定是一个坐标区间$[l, r]$。所以我们把凸多边形每条边变成两个点事件，在左端加入右端删除，维护当前竖线碰到的多边形的边即可。

88.4 时空复杂度

时间复杂度：$O(\text{坐标范围}+n\log n)$

空间复杂度：$O(n)$

89 White, Black and White Again

89.1 试题来源

Codeforces 306C

89.2 题目大意

Polycarpus的生活总是满足“一些好事，然后一些坏事，然后一些好事”这样的规律。所以Polycarpus认为接下来的n天也是满足这样的规律的。

Polycarpus知道，接下来会发生w件两两不同的好事和b件两两不同坏事，每天至少发生一件事，每天要么全部发生好事要么全部发生坏事。

由于Polycarpus的规律，这n天会先有若干天发生好事，再有若干天发生坏事，再有若干天发生好事。(若干代指> 0)

要求统计事件发生的方案数（每天发生的事的顺序也不一样），答案取模$10^9 + 9$输出。

$n, w, b \leq 4000$

89.3 算法分析

简单的组合计数。

我们把好事看做白点，坏事看做黑点，不同的事件只按好坏区分，最后再给答案乘上$w!b!$。

考虑这一坨黑点有多少个白点，一共有$(w-1)$种方法。

然后我们来插隔板分出每一天。注意一共有$(w+b-1)$个隔板，其中有两个隔板是必须选的（黑白交界处），一共要分成n天，即插入$n-1$个隔板，所以方案是$\binom{w+b-3}{n-3}$。

于是得到最终计算公式：$w!b!(w-1)\binom{w+b-3}{n-3}!$

89.4 时空复杂度

时间复杂度：$O(1)$

空间复杂度：$O(1)$

90 Buy One, Get One Free

90.1 试题来源

Codeforces 335F

90.2 题目大意

有n个蛋糕，第i个蛋糕售价a_i。

商店大发福利，买一送一，每一份蛋糕，你可以免费拿走一份价格严格小于你买的这份蛋糕的蛋糕。

求买走所有蛋糕的最小花费。

$n \leq 500000$

90.3 算法分析

先换个姿势，最大化省的钱，即最大化免的蛋糕的花费和。

我们把相同价格的蛋糕合并起来，得到若干二元组$(price_i, cnt_i)$，按价格从大到小排好序。

用一个堆Q存所有免的收益增量，每次新加入一个物品$(price, cnt)$，设之前的物品总数为sum。

首先令$cnt = \min(cnt, sum)$，那么有$\min(sum - 2 * Q.size(), cnt)$个新物品是可以直接免的，把这些价值为$price$增量存进数组$sv$。

设还剩下res个物品没有被免，那么我们两个一组打包加入，考虑Q中之前存的增量，如果拆掉一个增量tmp，那么收益会减少tmp，但是空出两次免的机会。于是我们每次从Q中取出最小的增量tmp，如果$price \geq tmp$，则把两个$price$增量放进sv，否则，我们把一个tmp增量放进sv，再把一个$2*price-tmp$放进sv。

最后，将所有sv中存着的新的增量塞进堆Q。

考虑复杂度，每次塞进堆中的新增量不超过cnt，所以一共至多插入$O(n)$个元素到堆中。

90.4 时空复杂度

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

91 Balance

91.1 试题来源

Codeforces 317C

91.2 题目大意

有n个容器，每个容器的最大容量为V。

有m条无向管道，每条管道连通两个容器。

初始时第i个容器里恰有a_i的水，我们希望最终容器i里恰有b_i的水。你可以每次通过一条管道运输水，但要保证任意时刻每个容器内水量非负且不超过V。

求一组可行方案，总步数不超过$2n^2$。

$n \le 300$

91.3 算法分析

首先，如果某个连通块内的总初始水量和总目标水量不同，那么问题无解。

否则，我们队每个连通块随便求一棵生成树。

考虑每一个缺水的点u，以u为根在其连通块里作一棵有根树，然后枚举一个水多余的点v，现在我们的目的是不改变其他容器最终水量的前提下，合法地把水从v运到u。

我们算出需要运的水w，分两种情况。

当$w \le \lfloor \frac{V}{2} \rfloor$时，定义$solve(v,w)$表示运水过程。

如果v父亲装得下就先$trans(v,fa_v,w)$再$solve(fa_v,w)$；

否则，先$solve(fa_v,w)$再$trans(v,fa_v,w)$。注意w不超过容量的一半，所以后面那个补救方案是可行的。

当$w > \lfloor \frac{V}{2} \rfloor$时，分两次运，转化为上面那种情况了。

91.4 时空复杂度

时间复杂度：$O(n^3)$

空间复杂度：$O(n^3)$

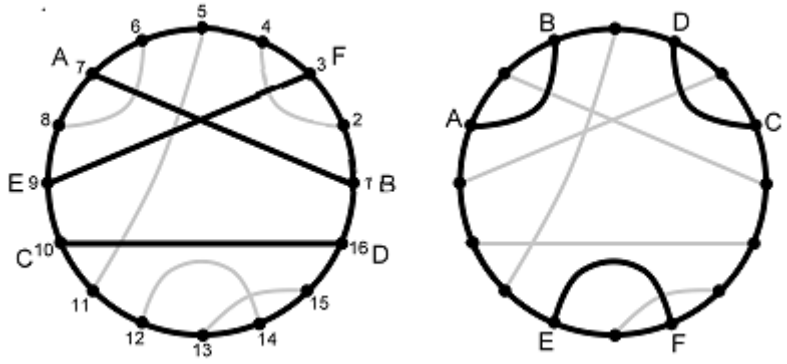
92 Mystic Carvings

92.1 试题来源

Codeforces 297E

92.2 题目大意

有一个$2n$个点的圆环，这些点连成了n个匹配。



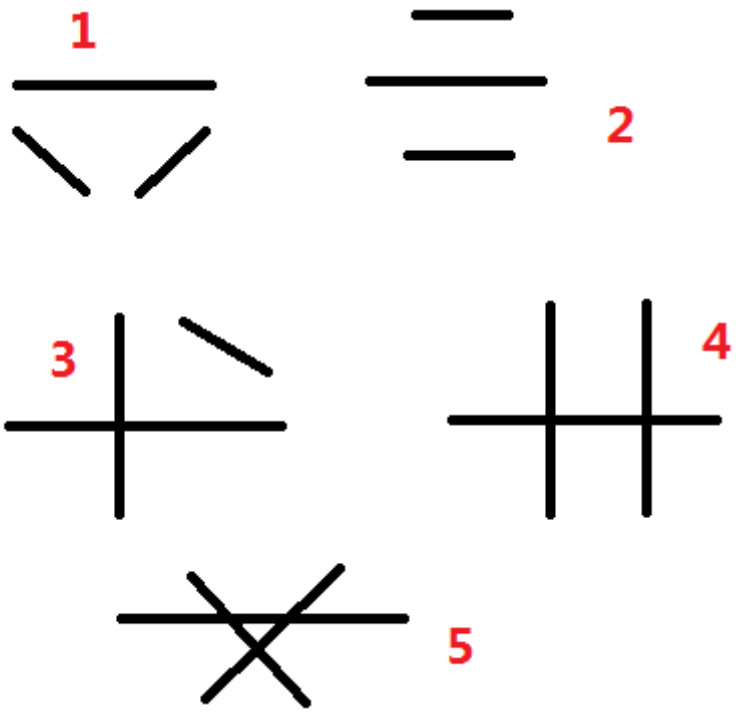
求有多少种选出三个匹配的方案，满足这三个匹配的跨度相等。

我们把选出的三个匹配的6个关键点标记出来，一个匹配(x,y)的跨度为从x在环上走到y需要经过的最少关键点数目。

$n \le 10^6$

92.3 算法分析

我们先分析三条线段一共有哪些情况。



如图，其中1和5是我们需要统计的情况。但是直接统计似乎比较困难，我们考虑反过来，统计2、3、4的情况。

首先我们用树状数组统计出每条线段左边的线段数L_i、右边的线段数R_i、相交的线段数M_i。

枚举一条线段i，再选一条相交的、选一条不相交的。这样34情况分别被统计了恰好两次。

枚举一条线段，左右各选一条，这样2情况被统计了恰好一次。

所以$ans = \binom{n}{3} - \frac{1}{2}\sum_{i=1}^{n} M_i(L_i + R_i) - \sum_{i=1}^{n} L_i R_i$。

92.4 时空复杂度

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

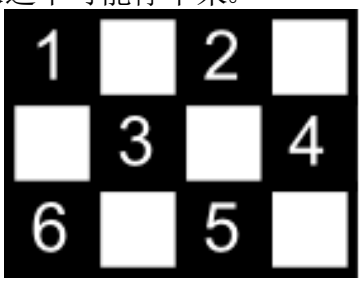
93 Shaass and Painter Robot

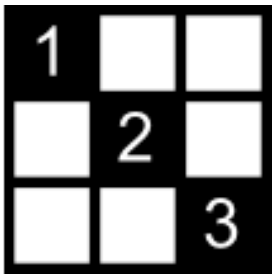
93.1 试题来源

Codeforces 294D

93.2 题目大意

给你一个$n*m$的网格，一开始都是白色的。上面有一个机器人，一开始位于格子(x,y)上（占据整个格子），面朝某个方向（左上，左下，右上，右下之一）。然后机器人会一直顺着这个方向走下去，每当遇到边界时会遵循光的反射定律改变方向，然后继续走。每当机器人走到一个格子后，它会将这个格子染黑，用掉一个单位颜料。即便这个格子已经被染黑了，也需要耗费一个单位颜料。当机器人意识到这个$n*m$的网格已经变成黑白相间的时候，它会立即停止行动。现在希望你求出，机器人停下来的时候，已经耗费了多少颜料？或者指出永远不可能停下来。





93.3 算法分析

为了把棋盘黑白染色，机器人至少需要到边界上的每个黑格子一次。

进一步地，可以证明，如果边界上对应的点都访问过了，整个棋盘就已经黑白染色完毕了。

首先我们把每个边界格子看做一个点，这些点之间可能的移动看作边，那么每个非角落的格子度数为2，每个角落里的格子度数为1。

考虑一下，一个连通图每个点度数都至多是2，那么要么是一个环要么是一条链。不论是哪一种，跑一遍之后每条边都将被经过至少一次。

所以考虑一个非边界格子，它被一条方向为左上的边穿过、一条方向为右下的边穿过，又因为这两条边都被经过了一次，所以这个格子必然会被染黑。

于是我们只需要模拟就行了，由于只关心边界的格子的染黑情况，最多模拟$O(n+m)$步即可。

93.4 时空复杂度

时间复杂度：$O(n+m)$

空间复杂度：$O(n+m)$

94 Ciel and Flipboard

94.1 试题来源

Codeforces 321D

94.2 题目大意

有一个$n \times n$的板子，每个格子上有一个数字（可能为负）。

n是一个奇数，令$x = (n+1)/2$。

你可以进行若干次操作，每次操作把一个$x \times x$的子矩阵乘上-1。

你的目标是最大化板子上的数字和。

$n \leq 33$

94.3 算法分析

我们把矩阵乘-1看做异或操作，考虑给定一个目标的01矩阵M，如何判定是否合法。

考虑个格子$A(i,j)$、$B(i,x)$、$C(i,j+x)$，最终局面$A \text{ xor } B \text{ xor } C$一定为0，因为每次操作都会影响$A,B,C$中偶数个格子。

这是对于一行，对于一列也是同样成立的。

这意味着，已知$M(1 \sim x, 1 \sim x)$，我们是可以递推出$M(1 \sim n, 1 \sim n)$的。然而是不是对于每一个$M(1 \sim x, 1 \sim x)$，一共2^{x^2}种状态，都是可以到达的呢。

用线性代数的知识可以证明这一点，大概来说就是原本的操作之间互相线性无关，所以一共有2^{x^2}种不同状态可以到达，那么相应的，上述2^{x^2}种状态中就不应该有非法状态。

至此我们可以得到这样一个算法。

枚举$M(x, 1 \sim x)$，于是就推出了$M(x, x+1 \sim n)$。矩阵被截成了上下两半，又因为第x行已经确定，那么$Mi+x, j$可以由$M_{i,j}$推出，所以把下半部分的值对应的放到上半部分的位置上去。

然后每一行是独立的，枚举一行i，再枚举$M_{i,x}$，接着对每个位置贪心就行了。

94.4 时空复杂度

时间复杂度：$O(2^{n/2} * n^2)$

空间复杂度：$O(n^2)$

95 The Last Hole

95.1 试题来源

Codeforces 274C

95.2 题目大意

平面里有n个圆，随着时间的流逝半径一起长大，在第t秒时所有圆的半径均为t。

我们把平面底色看做白色，圆看做黑色，那些有界的白色连通区域就叫做洞。比如图中红色圈出的区域。

请你求出最后一个洞消失的时间，或者输出-1 表示没有洞出现过。

95.3 算法分析

注意到每个洞最终将收束成一个洞点，我们可以先找出所有可能成为洞点的点。

一个洞点必然是到最近圆心最远的一个局部极优点，这让我们想到Voronoi图的顶点们。我们枚举每个圆心，然后把它到其他圆心的连线的中垂线们做半平面交，得到的交点即为这个圆心周围的Voronoi顶点。

接下来的任务是判定一个点是否可能成为洞点，即它在被覆盖前的那一瞬是否在某个洞中。

假设当前判定的点为p，我们先求出p首次被覆盖到的时间t，然后，我们把所有满足$\frac{dist_{i,j}}{2} < t$的点对i,j连边，边权是向量i到j相对于点p的转角。得到这样一张边数为n^2级别的图后，我们要求这个图中是否存在边权和非0的环。用带权并查集判。

95.4 时空复杂度

时间复杂度：$O(n^2 \log n + n^3)$

空间复杂度：$O(n^2)$

96 Have You Ever Heard About the Word?

96.1 试题来源

Codeforces 319D

96.2 题目大意

给一个字符串s，每次找到一个最短的最靠左的形如XX的子串，删掉一个X。

求做完所有操作后的字符串。

$|s| \leq 50000$

96.3 算法分析

注意到我们删除的串的长度一定是单调非降的。

这是因为，我们优先删除短的重复串，而删除长的重复串不会导致出现更短的重复串。理由很简单，考虑一个长$2l$的重复串XXY（Y是重复串后面的东西），删除后变成XY，如果存在更短的重复串，说X某个前缀和后缀相同，这意味着XX中间夹着一个更短的重复串，所以XX就不会被删去。

枚举一个l，如果我们能先判断出是否存在长度$2l$的重复串，再$O(|s|)$删除掉，那么就可以做到均摊复杂度$O(|s|\sqrt{|s|})$了。

现在问题的关键是如何快速判断是否存在长度$2l$的重复串。

我们在串中所有l的倍数的位置标记为关键点，设重复串为XX，那么X一定跨过了一个关键点。

所以我们枚举两个相邻的关键点u, v，求出这两个关键点向前、向后的LCP（二分+哈希），看看拼起来是否超过了l。这样判断长l的重复串是否出现的复杂度是$O(|s|/l * \log|s|)$的。

判定的均摊复杂度是$O(|s|\log^2|s|)$

96.4 时空复杂度

时间复杂度：$O(|s|\sqrt{|s|} + |s|\log^2|s|)$

空间复杂度：$O(|s|)$

97 Numbers

97.1 试题来源

Codeforces 241D

97.2 题目大意

给一排列$a_{1\sim n}$，求一非空子序列，满足：

1. 异或和为0；
2. 把所有数字的十进制表示按原位置从前往后写成一排，形成大的十进制数，这个数 mod p为0。

$n, p \leq 50000$，p是质数。

97.3 算法分析

有一个比较厉害的结论，对于$n = 31$的所有每个排列，有很大概率都存在解。

因为模质数意义下的乘法有着很大的随机性，所以2^{31}中选择里，期望有$2^{31}/p$种模p为0，这些种方案的异或和有1/32的概率是0。感受一下概率还是挺大的。

所以只考虑[0, 32)的数字，暴力DP即可。

97.4 时空复杂度

时间复杂度：$O(\min(n, 32)^2 * p)$

空间复杂度：$O(n)$

98 Sereja and Squares

98.1 试题来源

Codeforces 314E

98.2 题目大意

有25种括号，现在有n个位置，有些位置上已经填好了某种类型的左括号，求合法的括号序列的方案数。

$n \leq 10^5$

98.3 算法分析

首先不妨把所有括号看成一种，最后再乘上映射的方案数即可。

考虑怎么求括号序列呢？我们需要一个小常数的算法。

直接对这个数组维护：f_j当前填了j个右括号的方案数。

注意括号序列的充要条件是：把左右括号分别看做$+1,-1$，每个前缀和都非负，并且所有数的和为0。有了当前位置i和右括号个数j，就能推出左括号个数$i-j$，于是哪些状态合法就一目了然了。

直接暴力转移。

98.4 时空复杂度

时间复杂度：$O(n^2)$

空间复杂度：$O(n)$

99 Donkey and Stars

99.1 试题来源

Codeforces 249D

99.2 题目大意

直角坐标系内有n ($n \leq 10^5$)个点，定义点a能走到点b当且仅当：从a发出两条射线,和x轴正方形夹角分别为α_1、α_2，b严格在两条射线的内部。

求一条从原点出发的最长的路径。

99.3 算法分析

若a能走到b，设a出发的两条射线为l_1、l_2，则b在l_1的左侧，l_2的右侧。从线性规划的角度来看，每个限制条件都形如$A * x_a + B * y_a < A * x_b + B * y_b$。

现在模型变成了：每个点有两个属性，一个点可以移动到两个属性都严格大于自己的另一个点，求从原点出发的最长路。这是一个经典问题，可以按一种属性排序，再对另一种属性用树状数组做最长上升子序列。

99.4 时空复杂度

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

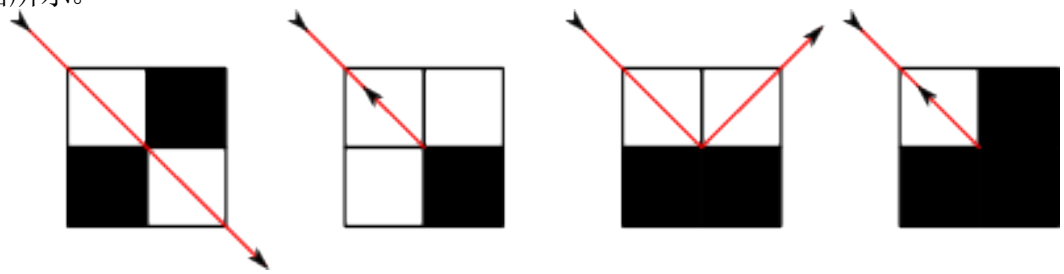
100 Mirror Room

100.1 试题来源

Codeforces 274E

100.2 题目大意

假设有一个$n \times m$的网格。左上角的格子坐标是$(1,1)$，右下角的格子坐标是(n,m)。网格中有k个堵塞的格子，其他的格子是空的。你在空格子(xs,ys)的中心向一个对角线方向（也就是东北，西北，东南，西南）发射一束激光。如果光束碰到堵塞的格子或网格边界，它会反射。在不同情况下光束的反射方式如下图所示。



过了一会儿，光束进入了一个无限的循环。计算至少被光束通过一次的空格子数。我们认为光束通过了一个格子的中心才算是通过了这个格子。

$n,m,k \le 10^5$

100.3 算法分析

我们先在网格的最外圈包一层堵塞格表示边界。

首先，一个格子是肯定不会被交叉穿过两次的。

考虑那些格点，把它们黑白染色，那么我们的光线一定是在同色的格点之间移动。注意一个格子，其两个对角线方向的格点颜色不同，所以至多只会从一个方向被穿过。

接下来考虑两种情况。

第一种情况，光线绕了一个大环。

第二种情况，光线碰到某个障碍物的角，原路返回。

注意每个格点的度数都是2，所以不会有其他的情况。

至此我们得到一个算法，把光线按原方向和反方向分别射出一次，统计回到起点所经过的格子数目之和。最后除以2得到答案。

首先，移动步数显然是$O(n+m+k)$的，这是状态量的级别，走到重复状态必然会形成环。

我们需要的是快速定位：从某个地方斜着射出碰到的第一个障碍物。

注意n,m都不大，于是我们对每一斜行都开一个*vector*记录该斜行上的障碍物，排好序，每次查询时在对应斜行里二分就能定位了。

100.4 时空复杂度

时间复杂度：$O((n+m+k)\log(n+m+k))$

空间复杂度：$O(n+m+k)$

101 Lights

101.1 试题来源

Google Code Jam 2009 World Final - F

101.2 题目大意

一个正方形的房间里有一个红光源、一个绿光源、n ($n \le 50$)个圆柱。光线沿直线传播并且会被墙壁和圆柱所吸收。

房间里有些区域没有光线到达，是黑色的。有些区域只有红光到达，是红色的。有些区域只有绿光到达，是绿色的。还有些区域既有红光到达也有绿光到达，是黄色的。

请你统计房间里四种颜色每一种的总面积分别是多少。（所求面积不包括圆柱所占的面积）

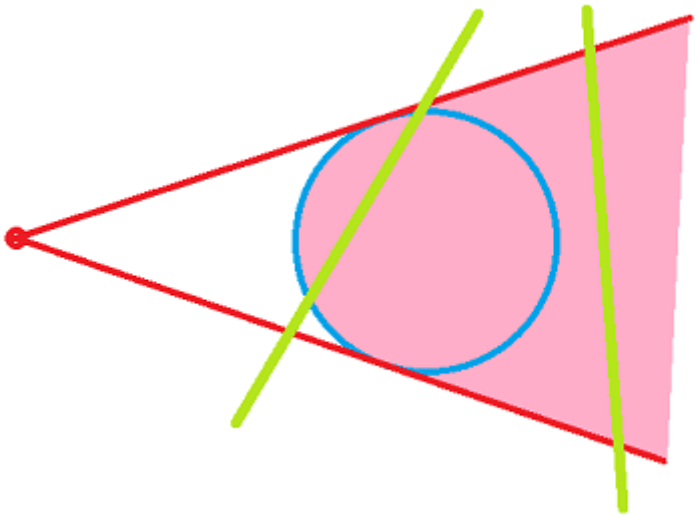
房间是$0 \le x, y \le 100$描述的区域，要求答案误差不超过10^{-5}。

101.3 算法分析

虽然题目所求有四个量，但如果我们知道了红、绿、黄的面积，黑的面积可以用总面积减去红、绿的面积再加上黄的面积得到。

由于坐标范围较小，精度要求不高，我们可以考虑积分，用辛普森公式来得到近似面积。想象一根竖线从$x = 0$ 扫描到$x = 100$，那么我们要求的就是对于某条竖线上：红光源照射的长度、绿光源照射的长度和两种光源都照射到的长度。经过一些分析不难发现，对于每个光源，每个圆柱所挡住的部分在这条竖线上都是一个区间。如果能得到这些区间，我们只需要做一遍区间覆盖问题，即：每个区间覆盖在左右两端标记点事件，对所有点事件排序，然后从 0 扫描到 100，然后途中进行所需要的统计即可。

现在问题的关键是：给定一个光源、一个圆柱、一条竖线，求圆柱挡的光在竖线上对应的区间。



我们先求出光源到圆柱的切线，那么就是要对于任意给定的绿线，回答其穿过粉色区域的区间段。这个看似简单的问题却似乎需要较繁琐的分类讨论。

我们换一个角度来思考——不用找到“准确的端点”，而是找到所有“可能成为端点”的点，然后再判断每一段是否可以照到。注意到这样的点只有：切线与竖线的交点、圆与竖线的交点。用这些点把竖线分成若干段后，每一段我们取中点来测试该段是否可以从光源照到（用点到线段的距离和圆柱半径比较）。

解决了这些问题，就能方便地用自适应辛普森来积分了。

101.4 时空复杂度

时间复杂度：$O(nK)$

空间复杂度：$O(n)$

102 Cow Neighborhoods

102.1 试题来源

USACO 2008 Open Gold

102.2 题目大意

平面内有n只奶牛，奶牛i坐标为(x_i, y_i)，如果两只奶牛的曼哈顿距离不超过C，我们给它们连上一条边。

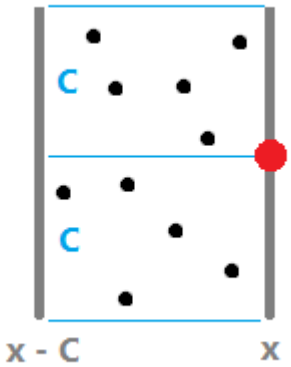
求得到的图的连通块个数、最大连通块的大小。

102.3 算法分析

为了方便，先进行坐标变换。把(x_i, y_i)变到$(x_i + y_i, x_i - y_i)$，这样i与j的曼哈顿距离就成了$max(x_i - x_j,\ y_i - y_j)$。

考虑这样一个暴力算法：把坐标变换后的点按x排序逐个加入，每加入一个点i时把所有满足$x_j \geq x_i - C$且$y_i - C \leq y_j \leq y_i + C$的$j$与$i$连边。这样做正确性是显然的，考虑如何优化。

对于x的限制我们可以维护一个集合，随着x_i的增加，一个个删除掉x小于$x_i - C$的元素们，再加入新的元素i，关键是连边复杂度较高。注意到我们的连边其实是有冗杂的。



我们发现，图中的两个方形区域内部的所有点必然是已经在同一连通块内的，因此i没必要一一向它们连边，向每个区域连一条边就足够了（可以考虑选

择最靠近边界的点）。

102.4 时空复杂度

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$