IOI2013中国国家集训队作业CODEFORCES题目泛做

石家庄市第二中学　毕克

2013 年 1 月 14 日

CodeForces ID: xlk

Contents

1 Volume I

1.1 7E Defining Macros

题目大意

使用C++的同学都知道：将x宏定义为成$y+z$之后，类似$x \times x$ 表达式会被解释为$y+z \times y+z$，而不是$(y+z) \times (y+z)$。本题给一些宏定义，最后再给出一个表达式，问有没有类似的问题。表达式长度≤ 120

算法讨论

对于一个表达式考虑这么几个值：

- 0表示这个表达式已经有问题了。
- 1表示这个表达式前面如果是$-$或着在前面或后面出现$\times$或$\div$就会出问题，是其他的则没问题。（即这个表达式内含有$+$或$-$）
- 2表示这个表达式前面如果是$\div$就会出问题，是其他的没事。（即这个表达式内含有$*$或/）
- 3表示这个表达式在前面是任何符号的情况下都不会有问题。

考虑这个上下文无关文法来确定计算顺序：

- $A \leftarrow A+B|A-B|B$
- $B \leftarrow B \div C|B*C|C$
- $C \leftarrow (A)|$变量

对于所有的没有被宏定义的变量和被括号包住的没有问题的表达式的值是3；对于所有运算如果一端是0那么这个运算返回0。对于每个运算来说：加法$+$对前后表达式没有任何要求返回1。减法$-$要求后面必须≥ 1，返回1。乘法$\times$要求前后表达式的值必须≥ 2，返回2。除法$\div$要求前面的≥ 2且后面的≥ 3，返回2 。

我们按照上下文无关文法求出这个表达式的运算优先级，并计算看最终结果是不是0即可。

本题应该有$O(length)$做法，即使用栈对表达式求值。但是考虑到表达式比较短没有必要。

这个做法时间复杂度为$O(length^2)$，这是我的做法，因为是递归求值代码很好写。

1.2 8D Two Friends

题目大意

有两个人X,Y和三个地点A,B,C，现在这两个人在A点，其中X要先到B点，再到C点，而Y要直接到C点。两个人速度一样，都是1单位路程/1单位时间。显然最近的路是直接直着走。，现在X可以在比最短路径慢t_1的前提下经过B到C点，Y可以在比最短路径慢t_2的前提下到C点。给出A,B,C的位置，和t_1,t_2。问两个人在第一次分开前最多共同待多久，分开后再走到一起不算，注意可以在A点附近逗留。

算法讨论

算法一 首先，我们特殊判断两个人可以一起走到B再一起走到C的情况，这样的话答案是$\min(\overline{AB}+\overline{BC}+t_1, \overline{AC}+t_2)$

然后我们考虑$\triangle ABC$，我们可以感觉到最优解是刚开始对着线段BC上的某点D走，走到一半分开。这个函数应该是一个凸函数。我们可以三分$(\overline{BD}/\overline{BC})$的值，然后计算最晚多久分开。

关于最晚多久分开这个问题。我们可以再用一个二分来做。首先我们特殊判断两个人可以一直走到D再分开的情况，然后我们假设在E点分开，二分$(\overline{AE}/\overline{AD})$，然后判断是否可行即可。

假设精度要求到$1/x$，（本题精度要求$x = 10^4$），时间复杂度便是$O(\log^2 x)$，可以接受，考虑到不需要做许多推倒代码也比较短，这是我的做法。

算法二 我们不妨换一个思路。首先第一部分特殊判断还是要做的。然后我们二分最终答案d，考察平面上哪些点可以作为分开的地点。我们注意到，这个点可以在的区间应该是三个圆的交集部分。具体来说就是以A为圆心，以d为半径的；以B为圆心，以$\overline{AB}+t_1-d$为半径的；以C为圆心，以$\overline{AC}+t_2-d$为半径的。我们看这三个圆有没有交即可。

这个的时间复杂度很优秀$O(\log x)$，但是需要做一个三圆求交的判定，相对较为麻烦。

1.3 8E Beads

题目大意

我们考虑长度为$n(n \leqslant 50)$的01串。将通过翻转或者是置换01 之后变成一样的分为同一组，每组的值是这组最小的那个01串，安值排序，求第$k(k \leqslant 10^16)$ 组的值。

算法讨论

这种题显然是每次暴力枚举下一位填0，然后算剩下还有多少组，如果还有很多就把0改成1，否则继续填下一位。

状态用$f_{i,j,k}$表示，i表示当前做到哪一位了，j表示从此之后还是否需要考虑里面的翻转，k表示是否需要考虑里面的置换。其中第一个表示做到哪一位了是两遍一起的。

至此本题做完，时间复杂度$O(n^2)$，这是我的做法，代码很短。

1.4 10E Greedy Change

题目大意

给定$n(1 \leq n \leq 400)$种价值的硬币$a_1, a_2, \ldots, a_n$ $(a_i \leq 10^9)$，保证$a_1 > a_2 > \ldots > a_n$,且$a_n = 1$。对于一个价格，有两种凑法：第一种是直接贪心，每次取不超过当前价格最大的面值来凑。第二种是使用尽可能少的硬币来凑。问最小要多少价格可以使贪心得不到最优解？（所谓优就是使用了更少的枚数，不存在输出-1。）

算法讨论

我们先给出一些的定义：

定义1.4.1. 一个硬币组是一个由n个非负整数组成的n维向量，本题中即为$A = (a_1, a_2, \ldots, a_n)$，在这里我们讨论的硬币组指的都是同一组。

对于一个硬币组，一个取法是一个由n个非负整数组成的n维向量，每个位置上的数分别表示取这种硬币的数量。定义一个取法$V = (v_1, v_2, \ldots, v_n)$的值为这个取法和他所属于的硬币组的内积，即$\Sigma_{i=1}^{n} v_i a_i$，定义一个取法的大小是所有位置上的数的和（即硬币个数），数学符号的形式即$V \cdot (1, 1, \cdots, 1)$。

注意我们的大面额在前。显而易见的，贪心就是取字典序最大的表示。

定义1.4.2. 我们定义两个取法$U = (u_1, u_2, \ldots, u_n)$和$V = (v_1, v_2, \ldots, v_n)$有$U < V$的关系，当他们满足对某个$1 \le i \le n$，有$u_i < v_i$且对于任意$1 \le j < i$，均满足$u_i = v_i$。

同时，我们为了方便表示，对于价格x，用第一种方案得到贪心取法为$G(x)$。由前所述$G(x)$为x的所有取法中字典序最大的。

我们考虑$x, y (x < y)$，显而易见$U = G(x) + (0, 0, \ldots, 0, y - x)$为$y$的一个取法。我们有$G(x) < U$，根据$G$函数的定义，还有$U \le G(y)$，所以$G(x) < G(y)$，接下来我们会看到，这是一个很棒的性质。

但是对于第二种方案的定义就不是那么容易了，因为尽可能少的硬币数的方案可能有很多。我们需要一个确定的对应关系。

定义1.4.3. x的最优取法，我们用$M(x)$表示，他是所有大小等于最优解的取法中，字典序最大的那个。我们定义硬币组A合理，当且仅当对于所有x，满足$G(x) = M(x)$。

对于两个向量U和V，我们说$U \subseteq V$，当且仅当，对于任意一维$i (1 \le i \le n)$，均有$u_i \le v_i$。接下来考虑两个有趣的结论。

引理1.4.1. 如果$U = G(U \cdot A)$，我们说U是贪婪的；如果$U = M(U \cdot A)$，我们说U是最优的。

1. 如果$U \subseteq V$且V是贪婪的，那么U是贪婪的。

2. 如果$U \subseteq V$且V是最优的，那么U是最优的。

下面给出证明。

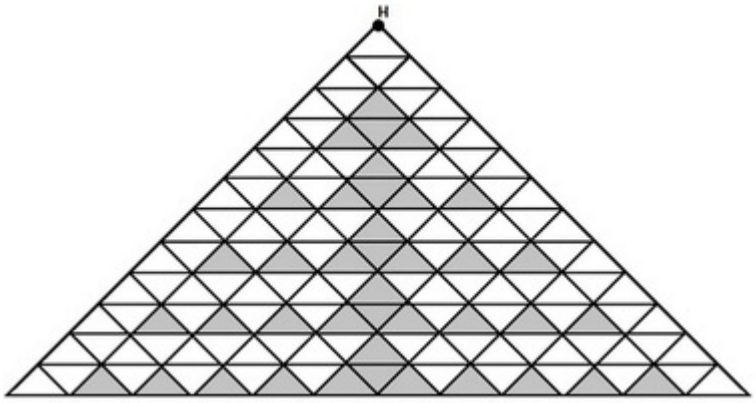
证明. 考虑第一个命题，如果存在更优的U'，那么他一定满足

$$U' \cdot A = U \cdot C$$

□

1.5 15E Triangles

题目大意



如上图所示，给出一个类似这样的图形，底边长是$n(n \leqslant 10^6)$。从H点出发，最终回到H点，任何一个边和点都只能经过一次（H除外），不能不走，求的路径条数，结果$\bmod 10^9+9$输出。

算法讨论

我们做一些标示方便描述。我们建立坐标系。第一维是平行于所有横线的，其中H点的第一维的坐标是0，每往下一条横线，坐标增加1。第二位是平行于从左上到右下的线，依然以H点坐标为0，每往左下一条线，坐标增加1。

我们注意到顺着和逆着的路径一一对应，我们只需要求出回路个数即可。我们注意到$(0,0),(1,1),(1,0)$经过这个回路的方案是特殊的，除了这个方案，剩下的方案的回路均包含$(1,1),(1,0),(2,1)$这个三角形。

然后我们讨论左边回路包住的区域有多少种情况，右边有多少种情况，显然这两个数目是相等的。下面我们以右边做讨论。

我们注意到这里有一些槽，这些槽进出一次就不能再经过第二次了。

比如从$(4,1)$进去，到$(5,1)$出来有5种方案（算直接从$(4,1)$到$(5,1)$），从$(6,1)$进去，到$(7,1)$出来有13种方案，事实上还有从$(2,1)$进去，到$(3,1)$出来有1种方案。我们设这个数列为a_i，有$a_i = 2a_{i-1}+3, a_1 = 1$。$a_i$就是当$n = 2i$的时候最下面的槽的方案数。为方便下面讨论，我们定义a_i的前缀积为$c_i = c_{i-1}a_i, c_1 = 1$。

接下来我们讨论另一个数列b_i，表示当$n = 2i$，从$(1,0)$到$(2,1)$ 的方案数，换句话说就是走过半边的方案数。我们只需要枚举有机会走进几个槽即可。有如下关系$b_i = b_{i-1}+4*c_i, b_1 = 2$。

所需要的最终答案就是$2b_i^2+2$，当$n = 2i$时。

运用这些递推关系即可完成本题目。至此，本题做完，时间复杂度$O(n)$，代码非常短。

1.6 17C Balance

题目大意

给定一个由abc组成的串，长度$l(l \leqslant 150)$，下面有两种操作。

1. 选两个相邻的字母，用第一个替换掉第二个。

2. 选两个相邻的字母，用第二个替换掉第一个。

定义平衡串为abc出现的次数至多差1，问最后可以生成多少不同的平衡串。

算法讨论

这显然是动态规划，我们定义$f_{i,j,k,l}$表示使用原串前i个字符，生成出了$i+j+k$长度的串，其中有i个a，j个b，k个c。转移的话只需要处理出来对于每个位置i，i及i之后第一次出现字母x的概率即可。

这样时间复杂度是n^4，所以状态要卡一下，明显没用的就不需要转移了（i,j,k的范围只有$\frac{l}{3}$）。

至此本题做完，时间复杂度$O(n^4)$，这是我的做法，代码很短。

1.7 17E Palisection

题目大意

给定一个字符串，长度$l(l \leqslant 2000000)$，求有多少对回文串相交。（自己和自己相交不算。）

算法讨论

显然总体思路是求出一共有多少对，然后有多少对不相交，然后相减。

求出一共有多少对，需要求出每个位置为中心的回文串有多少。这需要用到一个算法叫做Manacher 算法，不在赘述。

通过Manacher算法还可以一并求出，有多少个回文串以第i个字符为左（右）端点，我们处理出来这个，从前往后扫一遍即可。

至此，本题做完，时间复杂度$O(n)$，代码比较短。

1.8 19E Fairy

题目大意

给定一个无向图，$n(m \leq 10000)$个点，$m(m \leq 10000)$个边，去掉一个边使其变成二分图，输出删除哪些边可以达到要求。

算法讨论

二分图，就是没有奇环的图。

我们求出一棵dfs树，接下来我们围绕这个树做讨论。我们需要先做一个特殊判定，即如果原图就是二分图，我们删哪个边都可以。

我们注意到，每一个非树边对应一个环，而用这些环可以生成所有的环，所以我们只考虑非树边对应的环。我们计算有c非树边所对应的环是奇环。这里又需要一个特殊判定，如果$c=1$，我们可以选择删除非树边。

我们需要删除一条边，属于所有的奇环且不属于任何一个偶环。这个显然不会是非树边，因为非树边只属于一个环。

每个点我们维护一个f_i，我们把每个奇环对应的链（去掉非树边之后）的两端+1，两端的最近公共祖先的权值−2。（其实最近公共祖先一定是那个深度较浅的点。）这样之后，每个点所表示的子树中所有f_i的和即是有多少个奇环经过了这个点到他父亲的边。

偶环和奇环一样，也维护一个g_i

接下来我们统计多少点满足$f_i=c$且$g_i=0$，这些点到他们父亲的边就是所求。

这个题用到了一些奇怪的无向图的dfs树的知识。

至此，本题做完，时间复杂度$O(n+m)$，代码比较短。

1.9 23D Tetragon

题目大意

给定一个凸四边形三条长度相等的边上的中点。你的任务是还原出这个凸四边形。组数$T(T\leqslant 50000)$。

算法讨论

我们首先枚举逆时针依次是哪三个点，这一共有六种情况，暴力即可。

我们不妨事依次是A,B,C三个点，接下来我们讨论如果知道了这三个点，我们如何求出原四边形。我们设AB之间的顶点为E，BC之间的顶点为F。显然E在AB垂直平分线上，F在BC垂直平分线上，如果我们，作出A关于B对称点A'，作出E关于B对称点E'。我们注意到E'和F重合且是$\triangle A'BC$的外心。这样我们便可以求出F，再推出另外几个点。

小提示，如果使用复数类可以大幅度缩减代码长度。

算法复杂度是$O(T)$，代码比较短。

1.10 23E Tree

题目大意

给定一个有$n(n\leq 700)$个点的树，现在要去掉一些边，使得所有联通块的大小的乘积最大。求最大的乘积。

算法讨论

算法一 我们考虑定根之后做树形动态规划。状态$f_{i,j}$定义为以i为根的子树，i所在的点属于

的联通块有j个点，不算i所在的联通块的最大乘积，转移的时候类似背包。我们注意到每个点有$O(n)$ 的状态，转移复杂度为$O(n)$，所以时间复杂度是$O(n^3)$的，而且这还不算高精度。但是因为评测机的神速，和较小的常数（显然有很多没有用的状态），是可以过的。

算法二 我们先证明一个定理。

定理1.10.1. *存在一种最优解，使得最终结果中不存在长度超过4的链。*

证明. 我们假设最终结果中存在长度为$i(i \geqslant 4)$的链。我们将他分成2和$i-2$两部分，显然$2(i-2) \geqslant i$。 □

定理ref23E th1保证了最终每个联通块中只会有一个点的度数大于2。换句话说，就是每个点在最终的划分方案中，只有如下三种可能

- 他属于他父亲的联通块。
- 他是他属于的联通块中深度最浅的。
- 他是他属于的联通块的中心，即这个联通块内所有点都连向他。

这样我们给每个点定义三个状态x_i, y_i, z_i。x_i表示这个这个子树的最优解。y_i表示这个子树并上i这个点的父亲的最优解，z_i表示这个子树除去i之后的最优解。

关于转移，我们先讨论最简单的z_i，他的值就是所有x_j的积（其中j是i的孩子，下同）。然后我们讨论x_i。这个值的出现有两种可能，一种是i这个点被算入某个子树，这个比较简单，我们只需要找到y_j/x_j最大的子树，并与z_i相乘即可。还有一种情况，那就是i是这个子树的中心。这种情况我们需要先在遍历所有孩子的时候做一个动态规划，其中f_k表示在有k个孩子抛弃了他们自己的子树，来追随i点的最优解，转移是$f_k = \max(f_{k-1} * z_j, f_k * x_j)$。接下来$x_i$还要被$(k+1)f_k$更新一遍。$y_i$的时候和$x_i$的第二部分更新类似，只需被$(k+2)f_k$更新即可。

至此，本题做完，时间复杂度为$O(n^2)$（不算高精度）。只是需要熟练的书写高精度即可。

2 Volume II

2.1 26E Multithreading

题目大意

有$n(n \leqslant 100)$个线程，每个线程执行如下操作。

```
a[i].times{
    y[i]=x
    x=y[i]+1
}
```

其中$a_1, a_2, \ldots, a_n (1 \leqslant a_i \leqslant 1000)$给定。其中x是全局变量，就是说所有线程操作的是同一个x。你现在可以决定线程执行的顺序，就是说可以决定让哪个线程执行下一句话（给出的代码是两句）。通过决定顺序使得x最终为$w(-10^9 \leqslant w \leqslant 10^9)$。

如果存在方案，输出Yes和方案，否则输出No。方案是一些数，表示执行顺序，其中i出现$2a_i$次。

算法讨论

我们先讨论无解的情况，如果我们直接顺次执行n个线程，得到的是所有a_i的和。这是我们能得到的最大值，如果存在一个$a_i = 1$那么我们能得到的最小值是1，否则是2。如果$n = 1$，我们只能得到a_1。

通过这些我们去掉了所有无解的情况。然后我们特殊判定$n = 1$或者$w = 1$的情况，现在我们剩下的是$n > 1, w \geqslant 2$的情况。

我们构造如下方案，并证明他是正确的。

首先我们将$w = w - 2, a[1] = a[1] - 1, a[2] = a[2] - 1$。然后我们通过贪心得到数组$\{p_i\}$，满足$p_i$的和是$w$，且$p_i \leqslant a_i$。

我们构造如下方案。

1. 输出一个1。
2. 将$i(1 < i \leqslant n)$输出$2(a_i - p_i)$次。
3. 输出12。
4. 输出$2a_1$个1。
5. 输出2。
6. 将$i(1 \leqslant i \leqslant n)$输出$2p_i$次。

我们来说明这个方案是正确的。首先满足题目要求i出现$2a_i$次。其次，我们考虑算法执行过程，执完到第三步中的1之后，$\{y_i\}$数组里面有什么东西我们不去管它，我们注意到这时$x = 1$。然后执行到第五步。还是不管$\{y_i\}$数组是什么，这时$x = 2$。最后执行第六步，每执行两句话，x就增加1。

至此，本题做完，时间复杂度$O(n)$，代码比较短。

2.2 28D Don't fear, DravDe is kind

题目大意

给出$n(n \leqslant 10^5)$个东西排成一列，第i个东西有四个属性$v_i, c_i, l_i, r_i (1 \leqslant v_i \leqslant 10^4, 1 \leqslant c_i \leqslant 10^5, 1 \leqslant l_i \leqslant 10^5, 1 \leqslant r_i \leqslant 10^5)$，选出一个子序列$i_1, i_2, \ldots, i_k$。满足如下条件。

1. 对于子序列中第j个，满足j左边的所有东西的c_i和为l_i

2. 对于子序列中第j个，满足j右边的所有东西的c_i和为r_i

3. v_i和最大。

算法讨论

我们考虑动态规划。定义状态$g_{i,j}$表示，左边已经填好的c和为i，右边需要填的c和为j，满足这个条件的最优解的最右边是哪个。定义状态f_i表示以i作为最后一个的最优解的值。由于要输出方案，定义p_i为在f_i取最优解的情况下，第i个的前面是哪个。

还有一个问题没有解决，g的状态比较多，肯定会超内存。其实$g_{i,j}$只是辅助转移的。但是我们注意到实际使用到的不多，我们只需要写一个Hash即可（显然C++选手可以直接使用STL map）。

至此本题做完，时间复杂度$O(n)$或$O(n \lg n)$，取决于如何实现Hash。代码比较短。

2.3 30D King's Problem?

题目大意

给定x轴上n个点的横坐标$a_1, a_2, \ldots, a_n$和平面上一个点(x, y)，共$n+1$个点。给定k，求从点k出发的一条路径，经过所有点且距离最短。

算法讨论

我们先把x轴上的点排序。定义函数$D(i)$，为从第i个点到特殊点的距离。定义函数$A(l, r)$表示从那个特殊点出发，经过x轴上的第l到第r的点的最优解。定义函数$B(l, r)$表示从指定的起点出发，经过x轴上的第l到第r的点的，最终到达特殊点的最优解。

如果起点是特殊点的话，直接就返回$A(1, n)$即可。如果不是，我们考虑如下定理。

定理2.3.1. *在到达特殊点之前，一定到过第一个点或者最后一个点，*

证明. 假若都没有经过，这样的最优解是$D(k) + A(1, n)$。我们考虑从k到n到特殊点到$k-1$到1这条路径（如果$k = 1$那么就类似地先走左边再走右边）。我们将相同的部分约掉，有$D(k-1) \leqslant D(k) + a_k - a_{k-1}$，这样我们便构造出了更优解。 □

证明了这个我们就可以描述这条路径了，他是先走了一个区间这个区间包含a_1或者a_n，然后去了特殊点(x, y)，然后走完了剩下的区间。我们注意到第一个区间因为包含一个端点，所以数目是$O(n)$的区间数，我们只需要暴力枚举即可。

接下来我们考虑那三个函数，首先$D(i)$的实现很简单。我们考虑$A(l, r)$，他的结果应该是$a_r - a_l + \min(D(l), D(r))$，我们只需要到两个端点即可。我们考虑$B(l, r)$，他的结果应该是$a_r - a_l + \min(D(l) + \mathrm{abs}(a_k - a_r), D(r + \mathrm{abs}(a_k - a_l)))$因为最终要停在特殊点，所以一定要先把$x$轴上的点走完。

至此，本题做完，时间复杂度$O(n)$，代码比较短。

2.4 32E Hide-and-Seek

题目大意

A和B站在平面上，平面上有一面镜子，两个端点分别是M_1和M_2；还有一面墙，两个端点分别是W_1和W_2。给定这6个点的位置，问A能否看到B。

算法讨论

如果A要看到B只有两种光路，一种是直接看，一种是反射看。首先如果镜子挡住了视线，那么肯定看不到。否者我们看能不能直接看（即没有被墙挡住），如果能就一定可以看到了。如果不能我们讨论我们能不能看到B在镜子中的像B'，这个只需要保证，B'和A连线和镜子相交于C，然后AC和CB不和墙相交即可。

2.5 35E Parade

题目大意

在一个数轴上有$n(n \leq 100000)$个楼，问最后轮廓线是什么样的，每个楼的形式是一个区间$[l_i, r_i](-10^9 \leq l_i < r_i \leq 10^9)$和高度$h_i(1 \leq h_i \leq 10^9)$。

算法讨论

算法一 我们注意到，这是一个区间修改问题，我们可以用线段树做。先对所有的l_i, r_i离散化，然后进行每次操作，然后处理出来每段的高度，最后扫描一遍输出即可，时间复杂度$O(n \log n)$。这个做法可以AC，但是代码复杂度相对较高。本人没有实现此方法

算法二 我们进一步考虑题目条件，发现在修改过程中没有查询，利用这个特点，我们在离散化之后将每个楼按h_i升序排序，然后本问题转化为一个区间染色问题[1]。这样我们可以用并查集实现。最后也是扫描一遍输出，时间复杂度$O(n \log n)$。这是我的做法，个人认为最好，代码比较短，速度也比较快。

算法三 我们还可以从可以离线的角度考虑，我们可以记录每个位置x，有哪些区间从他开始或结束，这样我们从最左端向最右端扫描，中间用一个平衡树来维护最值。最后还是说扫描一遍输出，时间复杂度$O(n \log n)$。这个做法对于可以使用STL容器的C++选手很短，凭借CF神奇的评测机可以AC，但是常数较大，不建议使用。

2.6 36E Two paths

题目大意

给定一个无向图点数$n(n \leqslant 10^5)$，边数$m(\leqslant 10^5)$。找两个路径遍历所有的边。路径不能为空，输出方案。

[1]这个问题在2007年陈启峰的hw1给出了做法，原题为《疯狂的馒头》

算法讨论

欧拉路大家都会。

我们考虑如果联通块数大于2，无解。如果等于2，肯定一边一条路径，一下就是求两次欧拉路，下略。如果只有1，那么我们考虑有多少个奇点。如果4个以上无解。如果2 个或0个，我们直接求出一条欧拉路，然后拆成两段即可。如果4个，我们选两个奇点加上一个边，然后求一条欧拉路，然后按加的边拆成两段即可。

时间复杂度$O(n+m)$。

2.7 37E Trial for Chief

题目大意

给定一个$n \times m(n, m \leqslant 50)$的棋盘，每个格子是B或者W，每次可以选一个B或W的联通块，将他们都染成B或W。问至少多少次使得所有格子都是W。

算法讨论

将同一种颜色看成一个联通块，缩成一个点，相邻的两个联通块连上一条边，这样大抵就是求一个直径了（当然和直径两端是什么颜色有关系）。其实还有一个更暴力的方法，枚举每个点，看需要次数最多的B需要多少次才能染到。

时间复杂度$O(n^2m^2)$，就是一个BFS而已。

2.8 39C Moon Craters

题目大意

给出一些线段，选出尽量多的线段，选出来的任意两条线段要么相离（端点重合也算相离），要么一个包含另一个。保证没有完全相同的线段。线段条数$\leqslant 2000$。输出方案。

算法讨论

直接暴力动态规划就可以了，先离散化坐标。提前存下以i为左端点的都有那些。我们令$f_{i,j}$ 表示从i 到j最多能取多少个区间。$f_{i,j}$ 可以是$f_{i+1,j}$ 或者是选取一个以i为左端点的，且右端点小于j的，这个只要暴力就好了。最后看看有没有覆盖i到j的线段即可。

转移的时候记下如何转移的。这样可以输出方案。

时间复杂度$O(n^2)$，代码很短。

2.9 39A C*++ Calculations

题目大意

C*++语言和C++语言非常相似，然而C*++的程序有时会出现意想不到的结果。比如像这样的算术表达式：

- 表达式=基本式/(表达式+基本式)/(表达式-基本式)
- 基本式=增量/(系数*增量)
- 增量=(a++)/(++a)
- 系数=0/1/2/.../1000

如$5*a++-3*++a+a++$是合法的C*++表达式。

计算这样的表达式的值的方法：首先是每个基本式进行计算，然后按照正常的算术运算法则计算。如果一个基本式包含$a++$，则先进行乘法运算再使变量a权值+1；如果一个基本式包含$++a$，则先使变量a权值+1再进行乘法运算。

然而基本式可以按任意顺序计算，这就是为什么计算结果是完全无法预料的。

你的任务就是去找到最大的可能结果。

算法讨论

先把每一小部分项搞出来，这个可以C++暴力搞。当然如果会高端的东西可以用JAVA等自带正则的语言。

然后我们按照系数排序，系数一样的时候先++a后a++。

按照这个顺序计算即可，这个的正确性是显然的。

时间复杂度为$O(n)$，用了ruby或者python很短。

3 Volume III

3.1 39E What Has Dirichlet Got to Do with That?

题目大意

给定$a,b,n(a \leqslant 10000, b \leqslant 30, n \leqslant 10^9)$。每次可以让$a+=1$或者$b+=1$，谁让$a^b$超过了$n$谁输。当然可能平局。问最后结果。

算法讨论

我们将$a^2 > n$且$b = 1$的情况特殊判定解决掉。然后剩下的状态有限，直接记忆化搜索即可。

时间复杂度O(状态数)，大约10^6个状态。

代码很短。

3.2 40E Number Table

题目大意

一个$n \times m(n, m \leqslant 1000)$的表格，每个格子是$-1$或者1，每行的乘积和每列的乘积都是$-1$，其中$k(k < \max(n, m))$ 个格子是确定的。问剩下的有多少组解。

算法讨论

首先如果$(n+m) \mod 2 = 1$那么无解，因为整个表的乘积不能既是1又是−1。

我们不妨认为$n \geqslant m$。因为$k < n$这样n行中就一定有一行是空的了。我们考察剩下的$n-1$行，对于每一行；如果这一行没有空的，那么看他的乘积，如果是−1，那么答案不变，否者答案变成0；如果这一行有c个空格子，那么只考虑这一行有2^{c-1}种方案可以填。我们将$n-1$行的方案数都乘起来，然后考察那一个空行，空行需要保证每列都是−1。所以不能乱填。

至此本题做完，时间复杂度$O(k \lg n)$。代码很短。

3.3 43E Race

题目大意

现在Berland正在举行一场赛车比赛。比赛赛道可以看成一条s公里长的直线。有$n(n \leqslant 100)$辆赛车参加了这个比赛，他们同时从赛道的起点出发。我们已经知道每辆赛车的行为——在每一阶段中赛车的速度是恒定的。第i辆赛车的第j个阶段可以用数对$(v_{i,j}, t_{i,j})$表示，$v_{i,j}$表示第i辆赛车在第j个阶段的速度，单位是公里每小时，$t_{i,j}$表示第i辆赛车在第j个阶段的行驶时间，单位是小时。每个阶段按顺序给出，赛车将按照这些行驶。

你的任务就是统计这次比赛中出现了多少次超车，超车指一辆车行驶到了另一辆车的前面。所有超车都是在一瞬间完成的，也就是说不会存在两辆车在一段正长度的时间中并行的情况。可能存在同一时间发生多次超车的情况，此时每一个超车都必须被统计。两辆车在赛道的起点或终点相遇的情况不需要统计。

每辆车的阶段数$k(k \leqslant 100)$。

算法讨论

暴力枚举任意两个车，然后算每次速度变化的时候相对位置的改变。没有什么比这个题能更简单的了。

时间复杂度$O(n^2 k)$，代码很短。

3.4 44J Triminoes

题目大意

一个类似于国际象棋棋盘的$n \times m(n, m \leqslant 1000)$，被扣掉若干格子，然后问能不能用$1 \times 3$的多米诺骨牌覆盖，这个股骨牌的两端是白的中间是黑的。如果能输出方案（即把不同的骨牌用不同的字母染色，只需用abcd四个字母）。

算法讨论

看上去想搜索吧，但实际贪心就可以了。每次遇到黑格子，先看能不能上下放置，然后看能不能左右放置，如此即可。

生成方案的时候随便构造一下即可。

时间复杂度$O(nm)$，代码很短。

3.5 45G Prime Problem

题目大意

$1,\ldots,n$，一共$n(n \leq 6000)$个数，分成尽量少的若干组，使每组数的和都为素数，答案要输出方案。

算法讨论

设$s = n(n+1)/2$。

接下来我们分类讨论

- 如果s是质数那么直接分为一组即可。
- 如果s是偶数，那么我们一定可以将s拆成两个质数，问题的关键就是如何用这n个数表示其中一个质数，直接贪心就可以了。
- 如果s是奇数且$s-2$是质数，那么我们把2分一组，剩下的一组。
- 如果s是奇数且$s-2$不是质数，那么一定不可以分成两组，我们分三组，3单独分成一组，剩下的把$s-3$按分偶数的方法分成两组。

对于s为偶数的情况，事实是在这个数据范围内一定可以有一个其中一个质数小于等于n的方案。

至此本题做完，时间复杂度$O(n^2)$，这是我的做法，但是实际代码运行时间很短。

3.6 45E Director

题目大意

给定$2n(1 \leq n \leq 100)$个互不相同的非空字符串$a_1,\ldots,a_n,b_1,\ldots,b_n$，将前n个和后n个配对，要求：

1. 首字母相同的对数最多。
2. 满足第一个的前提下，将所有对连起来($a_1\ b_1\ a_2\ b_2$ 这样)字典序最小。

输出最终方案。

算法讨论

最多的对数通过统计以每个字母开始的单词数量很容易计算出来，接下来我们考虑如何让字典序最小。首先我们来考虑，不妨设最终答案是$\{a'_n\}$和$\{b'_n\}$。

定理3.6.1. *最终答案里$\{a_n\}$一定是升序的。*

证明. 我们不妨考虑最终答案里$\{a'_n\}$不是升序的，即存在$a'_i > a'_j$且$i < j$。我们交换a'_i, b'_i和a'_j, b'_j，现在首字母相同的对数没有变，而字典序变小了。所以最终答案里$\{a_n\}$一定是升序的。 □

通过定理10.6.1，我们得到$\{a'_n\}$便是$\{a_n\}$的排序，接下来我们要做的是为每个a_i找到其配对。

定理3.6.2. 前$i-1$个已经确定，b'_i一定是 不会使得最终对数减少的 字典序最小的b_i

证明. 假设不是。他和最小的交换，便得到了一个更优的解。 □

根据定理10.6.2，我们便得到一个算法：我们对$\{a_n\}$和$\{b_n\}$排序，然后对于每个a_i，我们从前向后扫描b_i选择第一个不使对数变少的（这里可以每次都暴力$O(n)$检验，也可做出一定分析之后$O(1)$ 解决）。

至此，本题可以以$O(n^2)$的时间复杂度（推荐，也是本人做法，代码复杂度较低），或者$O(n^3)$的时间复杂度通过。

3.7 39C Moon Craters

题目大意

给定两个串，还有n个操作，每个操作类似把x变成yz这样，求一个最短的串，使得这个串经过操作，既能得到第一个串也能得到第二个串。

算法讨论

暴力动态规划，对于串求出$f_{i,j,k}$，表示i到j，能否用k这个字母表示。这个做法很经典，链接

然后我们再做动态规划gi,j第一个串到i第二个串到j至少需要多少个字母。

时间复杂度$O(n^4)$。

3.8 51F Caterpillar

题目大意

给定一个$n(n \le 2000)$个点$m(m \le 10^5)$个边的无向图，可以对两个点执行合并操作，问最少需要几次合并操作就可让原图为毛毛虫？合并两个点x,y就是指新建一个点z，将所有连着x,y的边都连到z 上，如果有重边那么就不连。毛毛虫就是一种特殊的树，存在一条路径，使得所有点都在这个路径上或紧邻这个路径（即存在一个相邻点在这个路径上）。

算法讨论

考虑等价的限制条件，使最终的点数最多。

为了保证最终是一个树对于在同一个双连通分量内的点，一定要被合并到一起。这样我们得到一个森林。先考虑对于一个树最多剩下多少个点。我们注意到合并之后，可以保留所有的叶子，还可以保留一个最长链。最后我们考虑图不连通的情况，这样又需要联通块数−1的合并。

1. 我们求出双连通分量并缩点。

2. 我们对于剩下的每个树求一个直径，直径上的所有点和所有的叶子的总数是最终这个图能剩下的点的个数。

3. 为了保证图联通我们需要额外合并一些点。

整个算法复杂度为$O(n+m)$，这是我的做法，代码比较好写。

4 Volume IV

4.1 53E Dead Ends

题目大意

给定一个$n(n \leq 10)$个点m个边无向图，无重边自环。求有多少个不同的生成树满足有k个叶子。

算法讨论

如果直接状态压缩动态规划解决，$f[i][j]$表示用了i的点（二进制压缩）存在j的叶子（也是二进制压缩）。状态转移的时候只能是枚举新加了那个边。这么做有一个巨大的问题，就是同一个生成树因为最后一次加的边不同，而被计算两次。

为了避免这个问题，我们对转移做如下限制，每个树只能从去掉标号最大的叶子所连的边的状态转移过来，具体实现的时候就是枚举当前要加哪个边，然后看加上了这个边，新产生的这个叶子是不是标号最大的。

初始状态是一个生成树只含有一条边，最后统计答案只统计j的二进制表示中含有k个1的即可。

时间复杂度$O(3^n n^2)$，C++可以灵活的运用位运算进行判断和修改，这是我的做法，代码很短。

关于为什么是3^n而不是4^n作出解释，考虑到当$(i\ or\ j) \neq i$的时候一些状态是不存在的，我们考虑0到3^n-1所有的三进制数，显然一共有n位，我们用i 表示哪些位不是0，用j表示哪些位是2，这样便于我们所设计的状态一一对应了。

4.2 57D Journey

题目大意

给定$n*m(n \leq 1000, m \leq 1000)$的棋盘，每个格是空地或障碍物。保证没有两个障碍物在同一行或者同一列或者曼哈顿距离小于等于2（换句话说，不能对角相邻）。现在有一个东西，从任意一个空地走一个空地（两个空地均等概率随机产生，可能相同），每次移动可以向上下左右来移动，问每次移动步数的期望。

算法讨论

算期望一定是先算总路程是多少，然后除以总路程数。因为存在一些障碍物，直觉上是需要做一些绕行的。我们先算出来所有对点的曼哈顿距离之和。这个比较好做，我们只需要横竖分开讨论，先统计每行和每列有多少个空地，然后枚举是从哪里到哪里。并分别用$O(n^2)$和$O(m^2)$来计算即可。

引理4.2.1. *设起点坐标为$(0,0)$，终点坐标为(x,y)，不妨认为$x \geq 0, y \geq 0$，第i列的障碍物在(i,p_i)的位置上。需要绕的距离肯定不超过2。*

证明. 我们考虑数学归纳法。来进行证明

首先我们讨论$x=0$或$y=0$的情况，因为每行每列都只能有一个障碍物，且这个障碍周围没有其他的障碍物，所以一定可以通过多绕2的距离绕过去。

接下来我们假定$(x-1,y)$和$(x,y-1)$可以通过多绕2的距离通过，我们来看(x,y)是否可以。注意到$(x-1,y)$和$(x,y-1)$不能都是障碍物，我们选择没有障碍物的那一边走就可以了。

□

定理4.2.1. *不妨认为$x \leq y$（即x方向是较短的那边，否则交换x和y，因为较长的一边不可能满足这个条件），需要绕的时候当且仅当每列都有障碍物且$p_0 < p_1 < \ldots < p_x$。*

证明. 先证充分性，考虑所有障碍物和其上部若干格组成的区域，我们不可能仅仅通过向右和向上到达这部分区域。再证必要性。直接证明有困难，考虑否命题：如果存在一列没有障碍或者$p_i > p_{i+1}$，那么不需要绕。使用数学归纳法证明。

首先我们讨论$x=0$的情况，因为不能直接通过所以要绕。

接下来我们假定$(x-1,y)$和$(x,y-1)$满足定理。如果我们从走到$(0,1)$之后可以不绕，那么就走到$(0,1)$，否则说明满足$p_1 < p_2 < \ldots < p_x$这时，第0列一定是没有障碍或者是$p_0 > p_1$的，显然无论是是那种情况$(1,0)$都不会是障碍物。 □

根据定理10.6.1我们得到了强有力的判断方法。接下来只需要横竖（即对于两个点是x大还是y大）分开讨论，对于当前障碍物，向左和向右各扩张一次（就是在满足$p_i > p_{i-1}$ 的情况下一直往右的所有障碍物）然后求出这些障碍物上方区域的大小，并乘以当前障碍物下方的格子进行累加即可。

算法时间复杂度是$O(n^2+m^2)$，这是我的做法，虽然题解写了这么多但是大部分篇幅是证明，代码最终也比较短。

4.3 226E Noble Knight's Path

题目大意

给定一个树，每个点有颜色，颜色为黑或白。初始颜色为白。需要支持如下操作。

- 把一个点变成黑的。
- 询问两个点之间第y次操作之后到现在变成黑点的点之中的第k个点是哪个。

点数$n(n \leqslant 10^5)$，操作数$m \leqslant 10^5$。

算法讨论

显然是高级数据结构了。

先求出DFS序，然后我们建立一个线段树，因为需要维护历史信息，所以我们需要使用可持久化数据结构。

使用了可持久化数据结构之后便可以求出一个点到根的路径上有多少个点在y次操作之后变成黑点了。

然后我们每次找答案的时候直接倍增和判定即可。

时间复杂度$m \lg^2 n$，代码都是比较基本的数据结构，应该算比较好写。

4.4 217D Bitonix' Patrol

题目大意

有$n(\leqslant 1000)$个物件，和一个长度为$m(\leqslant 60)$的环（环上点的坐标为$0, 1, 2, \cdots, m-1$），第i 个物件有一个属性a_i，表示这个物件可以让你顺时针或逆时针移动这么远。

你要选定这n个东西一个子集，使得当你运动出去之后，你便不能回来了。

算法讨论

将所有a_i都对m取模，并变成$\min(a_i, m-a_i)$。显然重复的不能取两次，这样可以类比只剩下至多31个东西，然后我们用bitset暴力就可以了。

时间复杂度不好估计。事实是至多只有41541个情况。

4.5 67C Sequence of Balls

题目大意

给定两个字符串，长度$la, lb(la \leqslant 4000, lb \leqslant 4000)$。下面有四个操作，要通过这些操作使第一个变为第二个。

1. 加入一个任意字符，代价t_i。
2. 删除一个任意字符，代价t_d
3. 将任意一个位置上的字符换为任意字符，代价t_r。
4. 交换相邻的字母t_e

保证所有$t \leqslant 100$且$2t_e \geqslant t_i + t_d$。

算法讨论

显然是动态规划。

我们定义$f_{i,j}$为用第一个串的前i个字符，生成第二个串的前j个字符的最小代价。

我们发现交换操作不好考虑，我们注意到还有一个条件$2t_e \geqslant t_i + t_d$。这说明每个字符只会被交换至多一次，显然这个交换是一步到位的，就是说交换之后两个字符均各得其所，不需要再动了。

因此我们对两个串都预处理出$p_{s,i,j}$为对于串s从i个字符开始往前第一个出现j。

至此本题做完，时间复杂度$O(lalb)$，这是我的做法，代码很短。

4.6 70D Professor's task

题目大意

维护一个凸包支持两个操作。

- 加入一个点。
- 询问一个点在不在凸包内。

算法讨论

很经典的题了，想必大家都很熟悉做两个平衡树，分别维护上凸壳，和下凸壳即可。也可以只用一个。维护极角序。

至此本题做完，时间复杂度$O(n \lg n)$，这是我的做法。

5 Volume V

5.1 76F Tourist

题目大意

有一个人在X轴上行走，每个时刻的移动速度不超过V，它可以选择两个方向行走，也可以不动。他现在提前知道在t_i时刻x_i点会发生一个事件。

1. 从时刻0开始，从0点出发最多能参与的事件数量。
2. 从时刻0开始，从任意点出发，最多能参与的事件数量。

算法讨论

先考虑第二个问题。

能先后参加事件i和事件$j(T_i < T_j)$的条件是：

$$V(T_j - T_i) \geq |X_j - X_i|$$

由此可得，可以先后参加事件i和事件j（T_i与T_j的关系未知）的充要条件是：

$$\begin{cases} VT_j - X_j \geq VT_i - X_i \\ VT_j + X_j \geq VT_i + X_i \end{cases}$$

设$A_i = VT_i - X_i, B_i = VT_i + X_i$，那么问题等价于寻找一个最长的序列，满足元素$A$值和$B$值都单调不降。这个的做法是显然的，我们可以通过先对一维排序，另一位求LIS即可。

对于第一个问题，我们只需添加一个$A = 0, B = 0$的点，并强制让它成为最长不下降子序列的开头即可。

至此本题做完，时间复杂度为$O(n \lg n)$。

5.2 77E Martian Food

题目大意

在一个半径为R_1的圆A中，放一个与之内切的半径为R_2的圆B。接着连续放$(k+1)$个圆$C_0, C_1, \ldots, C_k$，每个圆要与圆A内切，与圆B、以及上一个放的圆（如果有的话）外切且半径尽量大。

保证$R > r, k \geqslant 1$。

多组数据，组数$T(T \leqslant 10000)$。

算法讨论

打表找规律，证明大概是用到了反演变换。把圆A的圆心放在$(R_1, 0)$，圆B的圆心放在$(R_2, 0)$。

算法复杂度$O(T)$，代码很短。

5.3 85E Guard Towers

题目大意

二维平面上有n个点。现在要对点进行分组，每个点要么属于A组，要么属于B组。

定义一个组的直径为这个组里面的所有点对间的曼哈顿距离中的最大值。

你要求某些分组，使得A、B两组的直径中的最大值最小。输出这个值，以及有多少种分组的方式。

算法讨论

因为是曼哈顿距离，对于每个点(x, y)，定义$(a, b) = (x + y, x - y)$。题目相当于：让你求一个最小的d，使得可以用两个边长为d的正方形覆盖所有点；还要求方案数（每个点被哪个正方形覆盖）。

显然有一个最优方案满足：要么一个正方形覆盖最左边和最下边的点，另一个覆盖最右边和最上边的；要么一个正方形覆盖最右边和最下边的“边界点”，另一个覆盖最左边和最上边的。（暂不考虑两个正方形是不同的）

两种情况下，分辨确定正方形的位置，便可以计算直径了。

计算的时候还需要考虑一个正方形同时覆盖了上下左右。

时间复杂度$O(n)$。

6 Volume VI

6.1 86E Long Sequence

题目大意

请你找出两个长度为$k(k \leqslant 50)$的01序列$A_1, A_2, \ldots, A_k$ 和$C_1, C_2, \ldots, C_k$，并对于$i > k$定义：

$$A_i = \sum_{1 \leqslant j \leqslant k} A_{i-j} C_j \pmod 2$$

设这个无穷序列A的最小循环节是m。你找的序列A和C要满足$m = 2^k - 1$。A和C分别不能全为0.

算法讨论

直接随机即可，解很多。

关于验证，对于i是m质因数，m/i均不是循环节。

时间复杂度不好确定，当然也可以先预处理出来然后打表。

6.2 91D Grocer's Problem

题目大意

给出一个n的排列P，每次操作你可以选出不超过5个数，对他们进行重新排列。求最少需要多少次操作，使得排列变为$\{1, 2, \ldots, n\}$。输出方案。

算法讨论

显然就是我们要调整一些置换环使得都是长度为1的环。

1. 拆所有长度大于等于4的环。
2. 拆2和3的环一对一对的。
3. 拆3环
4. 拆2环。

正确性显然，时间复杂度$O(n)$。

6.3 93D Flags

题目大意

一共有若干个格子，要给这些格子染色，每个格子有四种颜色可供选择，白黑红黄。要求如下：

1. 相邻两个颜色不能一样。

2. 白色和黄色不能连着。

3. 红色和黑色不能连着。

4. 不能出现连续三个是黑白红或红白黑。

可以通过翻转得到的算同一种方案，比如“黑白”和“白黑”算同一种方案，求格子数量在$[l,r](1 \le l \le r \le 10^9)$内的方案数和$\bmod\ 1,000,000,007$。

算法讨论

首先我们解决翻转的问题：设长度为n翻转算同一方案的方案数为a_n，翻转不算同一方案的方案数为b_n，满足条件的回文串方案数为c_n，我们得到如下结论

$$2a_n = b_n + c_n$$

我们来证明这个结论：

1. 对于任何一个回文串，对a_n贡献1，对b_n贡献1，对c_n贡献1。

2. 对于任何一个不是回文串的串和它的翻转，对a_n贡献1，对b_n贡献2，对c_n贡献0。

由此我们便可以得到上面的结论。

我们来观察c_n的特征：当n为偶数的时候c_n一定为0，因为中间两个必须是相同的颜色，但是相同的颜色不能相邻。当n为奇数的时候我们考察左半部分的选取，右半部分由左半部分对称得到而且一定是合法方案，左半部分的方案数为$b_{(n+1)/2}$。这样我们便完整的得到了如何用b_n表示a_n。

接下来我们只需讨论a_n的前缀和s_n和b_n的前缀和t_n的关系，最终答案便是$s_r - s_{l-1}$不难得到

$$s_n = t_n - t_{\lfloor (n+1)/2 \rfloor}$$

只要注意到我们要减掉的$b[i]$是连续的一段，这个结论还是比较显然的。

接下来我们来具体的求解b_n。设f_n为前n格，最后一格为黄色，设g_n为前n格，最后一个为红色(将状态定义为黑色也可以，这两个的方案数一定是相同的)，设h_n 为前n 格最后一格为白色，且倒数第二个为红色(将状态定义为黑色也可以，这两个的方案数一定是相同的。但是当$n = 1$的时候白色前面什么都没有，故$n = 1$总方案数是4，要特殊判定)。

接下来我们列出转移：

$$\begin{cases} f_n = 2g_{n-1} \\ g_n = f_{n-1} + h_{n-1} \\ h_n = g_{n-1} \\ b_n = f_n + 2g_n + 2h_n \end{cases}$$

化简之后我们可以的到一个关于b_n的表达式和一个g_n的递推：

$$\begin{cases} b_n = 2g_n + 4g_{n-1} \\ g_n = 3g_{n-2} \end{cases}$$

我们可以很轻易的通过枚举得到：$g_2 = 2, g_3 = 3, b_2 = 8, b_3 = 14$。我们便得到了$b_n$的通项：

$$b_n = \begin{cases} 0 & n = 0 \\ 4 & n = 1 \\ 14 \times 3^{(n-3)/2} & n\text{为大于1的奇数} \\ 8 \times 3^{(n-2)/2} & n\text{为偶数} \end{cases}$$

然后我们通过等比数列求和得到了一个关于t_n非常简洁的表达式。

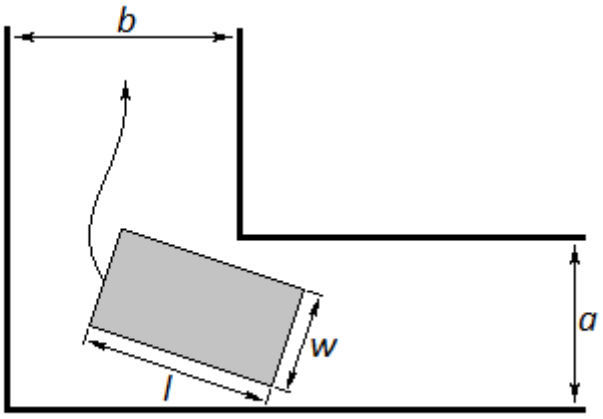
$$t_n = \begin{cases} 0 & n = 0 \\ 4 & n = 1 \\ 11 \times 3^{(n-1)/2} - 7 & n\text{为大于1的奇数} \\ 19 \times 3^{n/2-1} - 7 & n\text{为偶数} \end{cases}$$

至此我们可以得到$2s_n$，因为所有运算都要取模所以不能直接除以2，而要乘以2的乘法逆元。最终答案即为$s_r - s_{l-1}$

至此，本题以极其低的编程复杂度(仅需要一个快速幂)，和$O(n \log n)$的时间复杂度做完了，这是我的做法。其中部分地方比如递推过程可以用矩阵乘法暴力实现，也不失为一种降低思维难度的方法。

6.4 98C Help Greg the Dwarf

题目大意



如图，给定a, b, l，求最大的$w (w \leq l)$，使得矩形可以从右下移动到左上。矩形可以任意旋转、平移。

算法讨论

不妨设$a \leqslant b$，显然当$l \leqslant b$的时候，答案是$\min(a, l)$。下面讨论$w \leqslant a \leqslant b \leqslant l$的情况。

我们考虑这么移动，矩形一边贴着最下方，不断向左移；碰壁后，贴着墙壁往上提，然后竖直起来，再移动上去。也就是说，我们需要知道：一条长度为l的线段AB，一开始A在$(0,0)$，B在$(l,0)$；然后A不断上移，B不断左移，直到与Y轴重合为止；期间这条线段与点(b,a)的最近距离（如果(b,a)在线段AB的下方，那么距离为负）。这个最近距离就是最大的w（如果距离为负那么无解）。

设AB与X轴的夹角为α，显然α与此时线段到点的距离的关系是一个单峰函数，三分就可以了。

算法复杂度为$\lg eps$，代码很短。

6.5 191D Metro Scheme

题目大意

给出一个点仙人掌，点数$n(n \leqslant 100000)$，每次操作你可以选择一条简单路径或者一个简单环，覆盖上面所有的边。问最少多少次操作能把所有边覆盖恰好一次？

算法讨论

首先如果一个点的度数为2，我们可以把这个点缩掉。

对于剩下的点，设奇点数x，至少需要$x \div 2$个路径才可以。又因为有环，所以还要加上环数。

如此便可以通过。

时间复杂度$O(n)$，代码很短。

6.6 164D Minimum Diameter

题目大意

给定平面上的n个点。你需要去删掉恰好k个点$(k < n)$，使得剩下的$n-k$个点所构成的集合的直径尽量小。一个点集的直径是指集合中最远点对的距离。只含一个点的集合的直径等于0。

$k < n \leq 1000, k \leq 30$。

算法讨论

首先二分答案D。对于两个距离大于D的点A和B，我们在A和B之间连一条边。那么答案$\geq D$的条件就是这个图的最大独立集不小于$n-k$。暴力搜索即可。

时间复杂度不好计算，代码实际很快。

7 Volume VII

7.1 150E Freezing with Style

题目大意

给出一棵带边权的树，请找出一条长度在$[L,R]$的简单路径，使得该路径所包含的边权的中位数最大。点数不超过10^5。

算法讨论

首先二分答案x，对于边权小于x的边，边权重新赋值为-1；对于边权大于等于x 的边，边权重新赋值为1。那么问题转化为求一条边权和不小于0的、长度在$[L,R]$的路径。

接下来点分治即可，其中需要用到单调队列来维护。

总时间复杂度为$O(n\log^2 n)$。

7.2 101E Candies and Stones

题目大意

给定n和$x_1,x_2,\ldots,x_n$，m和$y_1,y_2,\ldots,y_m$。一开始$a=1,b=1$，每次操作可以选择给a或b加一。a不能超过n，b不超过m。定义权值和为所有达到的状态(a_0,b_0)的$(x_{a_0}+y_{b_0}) \bmod p$之和。$p$给定。$n,m\le 20000$。求最大权值和，并输出方案。

算法讨论

直接暴力就可以，不过不能保存方案。

我们考虑用压位的技巧，然后第一次做一半，第二次做一半。

这样就可以通过全部数据了。

时间复杂度$O(nm)$。

7.3 103E Buying Sets

题目大意

给定$n(n\leqslant 300)$个集合，每个集合的元素用1到n编号。要求选出其中某些集合，使得这些集合的并集的势，等于选出的集合的数目。

对于任意的$k(k\leqslant n)$，满足从中选出任意k个集合，这k个集合的并集的势一定大于等于k。

每个集合有一个权值，每个选择方案的代价是所选的集合的权值的和。

请输出代价最小的选择方案的代价。

算法讨论

把集合作为左部，集合的元素作为右部，如果集合a含有元素b，那么从左部的a 点连一条边到右部的b点。根据Halls定理，这个包含$2n$个结点的二分图一定有完备匹配。

求出这个匹配后，把与每个集合匹配的元素作为这个集合的代表元素。那么一旦选择了这个集合，这个集合里除了代表元素以外的元素所代表的集合也要被选取。这就是一个经典的最大权闭合子图模型了，用最小割即可。

7.4 107D Crime Management

题目大意

给出m个条件(C_i, M_i)，其中C_i是大写字母，M_i是正整数，且所有M_i的积不超过123。

请你找有多少个长度为n的只包含大写字母的字符串S，至少满足下列条件中的一个：

- S中C_1的个数是M_1的倍数；
- S中C_2的个数是M_2的倍数；

  ……

- S中C_m的个数是M_m的倍数。

C_i中可能有相同的字母，只需满足其中一个条件即可。注意C_i中没有出现的字母不可以出现在S中。

$n \leq 10^9, m \leq 1000$。

算法讨论

由于所有M_i的积不超过123，我们可以用一个不超过123的整数表示所有条件下当前字母数模C_i的值。用这个作为状态，矩阵乘法快速幂即可解决此题。

7.5 115D Unambiguous Arithmetic Expression

题目大意

我们定义UAE(unambiguous arithmetic expression)为：

1. 所有的自然数是UAE,有前导零的自然数（比如0000,0010）也是UAE；
2. 如果X和Y是UAE, 那么$(X)+(Y), (X)-(Y), (X)*(Y), (X)/(Y)$也是UAE；
3. 如果X是UAE, 那么$-(X)$和$+(X)$是UAE。

现在给你一个只包含0–9和$+, -, *, /$的字符串S，让你计算有多少种不同的UAE,满足去掉了括号符号后就变成了S。答案对1000003取模。串的长度不超过2000。

算法讨论

设$F[i][j]$为我们已经处理完S的前i个字符，还剩下j个左括号需要匹配的方案数，转移如下：

$$F[i][j] \to F[i][j+1]$$
$$F[i][j] \to F[i+1][j](S_{i+1}\text{是加号或减号})$$
$$F[i][j] \to F[i'][j-1](S_{i'}\text{为下一个操作符})$$

为什么是对的，留给读者思考。

7.6 120I Luck is in Numbers

题目大意

一对数字(x,y)的幸运度$f(x,y)$定义为：x和y打开的光条的数量的交集。比如$(1,6)$的幸运度是1，$(7,8)$的幸运度是3。

所谓光条就是一般的电子屏幕显示数字用的方法。

对于一个长度为$2n$的数字串S，它的幸运度定义为：

$$\sum_{1\leqslant i\leqslant n} f(S_i, S_{i+n})$$

现在给你一个数字串S，求字典序和幸运度都比S大的数字串中，字典序最小的一个。

算法讨论

这种题的做法实在是无聊。

先从大到小枚举$i(0 \leqslant i < n)$，表示答案串的前i位与S相同，然后枚举第$i+1$位的数字，并计算此时（确定了前$i+1$位）的最大幸运度。如果这个值比S的幸运度大，那么答案的前$(i+1)$位就确定了。从前往后一点点就可以了。

7.7 123E Maze

题目大意

一个树形迷宫，到每个点会等概率的走向一个没走过的点。给定每个点作为起点和终点的概率，求期望路径长度。

算法讨论

我们考虑从终点到起点的过程，其中每个分叉都有50%的可能进入。然后我们只需要DFS一遍即可。

时间复杂度$O(n)$。

8 Volume VIII

8.1 193E Fibonacci Number

题目大意

令F_i表示Fibonacci Number。求最小的i，使得$F_i \mod 10^{13} = x$ 输入x。

算法讨论

我们注意到Fibonacci Number关于10^i的余数是有周期的。然后我们只需要一位一位的暴力即可。

时间复杂度不好估计，如果用python写很短。

8.2 132E Bits of merry old England

题目大意

你被要求输出一个由n个数($n \leq 250$) 组成的序列p。有m个变量($m \leq 26$)给你用，变量必须是单个小写英文字母。

你被要求给出一个操作序列，输出这n个数。你能做的操作只有2种：

- 对某个变量赋值（格式: $variable = integer$。例如：$a = 5$）。
- 输出某个变量（格式：$print(variable)$。例如：$print(a)$）。

变量没有初值，也就是，你必须给变量赋过值才能输出这个变量。

对于某个你给出的操作序列，定义其代价如下：

- 每个赋值操作的代价是你想要赋给这个变量的值的二进制表示中1的个数。
- 每个输出操作的代价是0。

你被要求给出一个操作序列以输出这n个数。同时，必须最小化操作序列的代价。

输出这个最小代价，以及能达到最小代价的满足条件的操作序列。如果有多个操作序列都具有最小的代价，输出任意一个即可。

算法讨论

题目虽然很长，但是就是用不超过m条路径覆盖n个点恰好一次。费用流即可。

建图如下：

- 设源点为s，汇点为t。另设辅助结点a。再建$2n$个结点；
- s到a一条流量为m费用为0的边；
- a到$2i$一条流量1费用为p_i的代价的边；

- $2i+1$到t一条流量1费用0的边；
- $2i$到$2i+1$一条流量1费用无穷小的边；
- $2i+1$到$2j(j>i)$一条流量为1的边；如果$p_i=p_j$费用为0，否则费用为p_j的代价。

至此本做完，复杂度为费用流复杂度。

8.3 140F New Year Snowflake

题目大意

给定平面上无重复的$n(n \le 200000)$个整点（坐标的绝对值小于$5 \cdot 10^8$），保证没有重复，你可以至多插入$k(k \le 10)$ 个点，可以不插入。使得这些点是中心对称。问一共有多少个不同的对称中心，如果有无限多个对称中心输出-1。

算法讨论

我们先讨论如何验证一些点是中心对称的和如何求对称中心，不妨设这些点为p_m，下标从零开始。

我们只需要将所有点排序，如果所有p_i和p_{m-1-i}的中点都重合，那么他们是中心对称的，对称中心就是p_0 和p_{m-1}的中点。

我们先讨论无解的情况，这种情况下。每对p_i和p_{m-1-i}中都有一个是插入的。然后对称中心就有无限多个了。显然$k >= n$属于这种情况。

接下来讨论$k < n$的情况，我们注意到k的范围很小，我们考虑枚举对称中心。我们注意到最终的点集p_i和p_{m-1-i}其中$0 \le i \le k$，这$k+1$对中一定有一对的两个点都来自于之前的n个点，然后我们考虑枚举这个点，一共有k^2个可能（排序之后，前k个点和后k个点配对。）

考虑已知对称中心，然后如何验证。我们只需要看有多少个点没有和自己配对的点，这个数目就是需要插入的点。C++ 选手可以直接STL set来确定一个点是否存在。如果没有STL的支持。就只好二分查找了（枚举之前，已经排序了）。

综上所述，时间复杂度为$O(k^2 n \lg n)$，这是我的做法。

最后验证的时候，可以用两个指针来维护，这样验证就是$O(n)$的了。

8.4 152D Frames

题目大意

在一个$n \times m(n,m \leqslant 1000)$的格子图里，画了两个边长都不小于3的矩形边框。现在给出这个图，识别出这两个矩形的对角顶点，可能无解。

算法讨论

考虑所有可以做横向边的纵坐标和所有可以做纵向边的横坐标，我们只留下最大的两个和最小的两个。

然后暴力枚举端点即可。

算法复杂度$O(nm)$。

8.5 183D T-shirt

题目大意

人从1到n编号。你希望给每个人一件衬衫。不幸的是你不知道每个人适合的衬衫尺寸。有m个不同的尺寸，从1到m标号，每个人适合且仅适合一个尺寸。

你不知道人们的具体尺寸，所以你去问你的朋友Gerald。可惜的是，他也无法获知确切的尺寸，但他成功地获取了对于人i和所有的尺寸j，适合人i的衬衫尺寸为j的概率$P[i][j]$。

由于你计划给所有的人一件衬衫，你打算带上恰好n件衬衫。对于这n件衬衫，你可以携带尺寸的任意组合（你也可以带同一尺寸的多件衬衫）。当决定带哪些尺寸时，你不知道对于每个人的衬衫尺寸，所以你不得不只基于Gerald朋友给你的概率挑选尺寸组合。

你的任务是使得到自己衬衫尺寸的人数量的期望值最大。

更形式化地说。当你最终到达办公室时，你将会询问每个人他的衬衫尺寸。然后如果你仍然有那个尺寸的衬衫，你将会给他其中的一件。否则你不会给他衬衫。你将会按照次序询问人们，从1号人开始，然后是2号人，直到n号人。

算法讨论

设$A_{i,j}$为至少有i个人的尺寸是j的概率。答案显然是所有$A_{i,j}$中最大的n个数之和。

设辅助状态$F_{k,i,j}$表示，只考虑前k个人的情况下，至少有i个人的尺寸是j的概率。递推公式是：

$$F_{k,i,j} = F_{k-1,i-1,j} \cdot P_{k,j} + F_{k-1,i,j} \cdot (1 - P_{k,j})$$

这个算法是$O(n^2m)$的，而且瓶颈在于状态。

观察到，如果对于某个j，我们选择了$A_{i,j}$中的x个值，那么一定是$A_{1\ldots x,j}$，因为$A_{i,j}$在i上是单调不增的。而我们只需前n大的值。那么我们不需要预先计算出所有的A值和F值：一开始先算出$A_{1,j}$，之后每次从m个数中选出最大值加进答案；把$A_{i,j}$加进答案后，算出$A_{i+1,j}$来代替它；每次计算时需要$O(n)$计算新的F值。

时间复杂度$O(n(n+m))$。

8.6 217E Alien DNA

题目大意

给出一个由ACGT组成的字符串s，你需要执行n次操作，每次操作给出$l \leqslant r$，拷贝字符串中第l位到第r位出来，对他们进行错位排序，然后粘在原子串$S[l \ldots r]$的后面。所谓错位排序，就是先取出偶数位的放在前面，再取出奇数位的放在后面。

只需求出s的前k位。

$n \leqslant 5000, \quad |S|, k \leqslant 3000000, \quad l \leqslant r \leqslant 10^9$。

算法讨论

考虑反过来做。假设我们现在已经求出了原串的前k位，现在要还原(l,r)这个操作。如果我们求出了l到r上的字符，那么$r+1$到$2r-l+1$上的也可以知道。那么我们逐一遍历第$r+1$到$2r-l+1$位，它们的字符就是第l到r位上对应的字符，我们记录下来（如果超出了k那么不要处理）。这时候把$r+1$到$2r-l+1$删去。

由于遍历了多少就删去了多少，总共遍历的次数是$O(k)$的。

总复杂度是$O(k \lg n)$的。

8.7 135E Weak Subsequence

题目大意

对于两个字符串a,s，我们定义a是s的弱子序列：

1. s中存在一个子串等于a。

2. s中存在一个不是子串的子序列等于a。

现在问字符集大小（就是可以出现的字符种类数）是$k(1 \le k \le 10^6)$，最长弱子序列长度为$w(2 \le w \le 10^9)$，的字符串有多少个。

下面给出子串和子序列的定义，所谓s的子串，即把字符串看成数组$s_1, s_2, \ldots, s_n$，有$i,j(1 \le i \le j \le n)$，$s_i, s_i+1, \ldots, s_j$为s的一个子串，下文我们用$s_{l,r}$ 来表示。所谓s的子序列，即把字符串看成数组$s_1, s_2, \ldots, s_n$，有一数组$a_1, a_2, \ldots, a_m$，满足$1 \le a_1 < a_2 < \ldots < a_m \le n$，则$s_{a_1}, s_{a_2}, \ldots, s_{a_m}$ 为s的子序列。

算法讨论

这个题目直接入手有困难。我们不妨考虑另一个问题：最长弱子序列长度小于w，的字符串有多少个。这样既可以把条件转化为这个串中不存在长度为w 的弱子序列，又可以推回原问题（只需要将$w+1$的答案和w的答案相减即可）。

我们有如下定理和证明：

定理8.7.1. *当且仅当一个串去掉前$w-1$个字母之后剩下的部分中没有两个相同的字母且去掉去掉后$w-1$个字母之后剩下的部分中没有两个相同的字母，这个串不含有长度为w的弱子序列。换句话说，当且仅当在$s_{w,n}$和$s_{1,n-w+1}$中，均没有任何一个字符出现两次，这个串不含有长度为w的弱子序列。*

证明. 充分性：假设在$s_{w,n}$中有一个字符出现两次，位置设为$x,y(w \le x < y \le n)$，那么s 的子串$s_{x-w+1,x}$等于不是s子串的子序列$s_{x-w+1,x-1} + s_y$。在另一部分出现两次的证明类似。

必要性：假设存在长度为w一个子串和一个不是子串的子序列相等，考虑他们的第一个字母和最后一个字母，一定有一个字母两个字符串的位置不一样（不然那个子序列就是子串了）。不妨设是最后一个字母的那个位置不一样，然后我们注意到，这两个位置一定是属于$s_{w,n}$，但是这个位置上的字符出现了两次，与条件矛盾。 □

有了定理10.6.1，我们就可以断定最终的长度n满足$0 \le n < w + k$，其中$n \le w$是一个等差数列（当然也可以不计算，因为最终答案是将$w + 1$ 的答案和w 的答案相减，那部分抵消了）。对于后半部分我们就可以用乘法原理来计算这个问题了。但是还是需要分类讨论一下。

首先我们考虑比较简单的情况$s_{w,n}$ 和$s_{1,n-w+1}$没有交集，即$2w - 2 \ge n$。这样中间的部分不属于任何一个部分，我们直接算一个次幂就可以了两边是两个排列问题，相乘即可，最终答案是$k^{2w-2-n} \times (P_k^{n+1-w})^2$

其次我们考虑两个有交集。这样的话首先交集部分不能自己和自己重复，然后两边排列不能和中间重复。最终的表达式为$P_k^{n+2-2w} \times (P_{k+2w-n-2}^{w-1})^2$

然后我们注意到。相邻的两个长度之间是有关系的。可以在预处理1到k的乘法逆元之后用$O(1)$的时间推出下一个，总时间复杂度为$O(n)$。这是我的做法，代码较短，速度较快。

9 Volume IX

9.1 163D Large Refrigerator

题目大意

给出V，找出一组整数解a, b, c使得$abc = V$，并且$S = 2(ab + bc + ac)$最小。V以质因数分解的形式给出。$V \leqslant 10^{18}$。

算法讨论

不妨规定$a \leqslant b \leqslant c$，然后枚举$a$，显然要保证$a^3 \leqslant V$。枚举了$a$后，相当于要求$S = a(b + c) + \frac{V}{a}(bc = \frac{V}{a})$最小，那么我们继续枚举$b$。然后枚举$a$之后估计一下剩余的最优解是多少。如果比当前答案差就直接退掉。

这个时间复杂度不好估计，实际通过的较快。

9.2 167E Wizards and Bets

题目大意

两个人画了一张有n个点m条边的有向无环图（图中顶点由1编号到n）。称没有入边的点为源，没有出边的点为汇。

注意一个孤立点既是源也是汇。在这张图当中，源和汇的数目是相同的。

他们把源们按照顶点编号升序重新编号为1到k，把汇们同样编号为1到k。

为了进行赌博，这两个巫师发动咒语，从图中选出k条路径，分别从某个源出发到达某个汇。每个源汇都恰被一条路径覆盖一次，任何路径不在顶点处相交。假设终止于i号汇的路径是由源a_i出发的，我们称汇对(i, j)是一个逆序对当且仅当$i < j$且$a_i > a_j$。如果所有汇对$(i, j)(1 \leqslant i < j \leqslant k)$中的逆序对总数是偶数，那么第一个巫师赢得这次赌博，后者必须给他一块钱，否则第二个巫师赢，获得一块钱。

这两个巫师对赌博是如此地狂热，以至于他们不断地念出咒语来产生新的路径集合。终于，所有可能的路径集合都被产生过恰好一次。两个路径集合被认为是不同的，当且仅当存在一条边，出现在前一个集合的某条路径中，而在后一个集合中不出现。

为了兑现赌注，请你算出第一个巫师的净收入，答案模一个素数P。

算法讨论

我们注意到任何路径不在顶点处相交这个条件没有用，因为如果相交的话净收入是0。

预处理出每一个源点s_i到汇点t_j 的路径数$F_{i,j}$。那么答案就是：

说到逆序对就不得不提行列式，显然最终的答案就是F矩阵的行列式的值。求行列式的方法很简单，使用高斯消元使F化为倒三角的矩阵后（对于$i > j$，满足$F_{i,j} = 0$），主对角线上的元素的乘积就是答案。

9.3 232D Fence

题目大意

给定数列$a_i(1 < i \leqslant n, n \leqslant 10^5)$。每次询问一个区间$[L, R]$问有多少不与他相交的区间$[l, r]$满足$a_{L+i} + a_{l+i} = a_L + a_l$。

算法讨论

先做差分，然后求后缀数组，然后整个问题变为了在height数组里，离线处理所有询问，相当于问一个区间内大于一个数的数有多少个，这个可以很简单的离线处理。

时间复杂度$O((n + q) \lg n)$。

9.4 175E Power Defence

题目大意

有一只怪物，以一单位长度每秒的速度从$(-\infty, 0)$走到$(\infty, 0)$。你可以在$(x, 1)$或$(x, -1)$上修防护塔（x是整数），同一点上至多只能修一座塔。

塔有A、B、C三种，都有其作用范围以及数量限制。A和B是攻击型的，每秒对范围内的怪物造成固定的伤害。C是冷冻型的。如果怪物在k座C塔的作用范围内，那么速度降为$\frac{1}{k+1}$。

求对怪物可能造成的最大伤害。

算法讨论

显然防护塔应该紧靠在一起。

然后我们枚举哪些位置是减速塔，按照可以输出的时间来放攻击塔。

时间复杂度$O(2^n n \lg n)$。

9.5 178F Representative Sampling

题目大意

给定n个串，选出k个串，使得这k个串之间两两LCP之和最大。

算法讨论

先排序，求出相邻两个的LCP，然后我们考察最小的LCP的位置，比如是第i个和第$i+1$个的之间的LCP，我们将整个问题划分成两部分显然属于不同的两部分的LCP就是第i个和第$i+1$个的之间的LCP，这样递归下去。对于每部分同时维护$k+1$个状态，状态j，表示取j个的最优解。

时间复杂度$O(n^2)$。

9.6 178E The Beaver's Problem II

题目大意

给一个图片，问多少圆和多少正方形，保证圆和正方形不相交，保证肉眼可以识别。

算法讨论

我们先BFS出有多少比较大联通块，然后对于每个联通块找一些特征值，比如所有点到中心距离的平均值除以中心到最远点的距离之类的，然后我们通过积分算出正方形和圆的特征值，我们找一个两者之间的值，来区别这两个图形。

我找的是所有点到中心距离的平方的平均值除以最远点的距离的平方。这样的话如果是圆就应该是0.5，否者是0.3333。

时间复杂度$O(size)$。

9.7 180B Divisibility Rules

题目大意

Vasya在学校学习整除的规则。以下是其中的一部分。

- 能被2整除的数的规则：一个数能被2整除当且仅当它的最后一位能被2整除，或者换句话说，是偶数。
- 能被3整除的数的规则：一个数能被3整除当且仅当它的各个数位上的数字之和能被3整除。
- 能被4整除的数的规则：一个数能被4整除当且仅当它的最后两位形成的数能被4整除。
- 能被5整除的数的规则：一个数能被5整除当且仅当它的最后一位是5或0。
- 能被6整除的数的规则：一个数能被6整除当且仅当它能同时被2和3整除（也就是说，当且仅当它的最后一位是偶数而且各个数位上的数字之和能被3整除）
- 能被7整除的数的规则：Vasya不知道这个整除性质。

- 能被8整除的数的规则：一个数能被8整除当且仅当它的最后三位形成的数能被8整除。
- 能被9整除的数的规则：一个数能被9整除当且仅当它的各个数位上的数字之和能被9整除。
- 能被10整除的数的规则：一个数能被10整除当且仅当它的最后一位是0。
- 能被11整除的数的规则：一个数能被11整除当且仅当它的奇数位上的数字之和等于它的偶数位上的数字之和，或者它们相差一个能被11整除的数。

Vasya很感兴趣的是，有些整除规则十分类似。事实上，去检查一个数能否被2,4,5,8,10整除的时候只需要检查它的最后一位或几位是否满足某个条件。Vasya 把这种规则称为2类型规则(2-type)。　　如果检查一个数能否被给定的数整除意味着计算这个数的各个数位上的数字之和并判断这个和能否被给定的数整除，那么Vasya称这个规则为3类型规则（3-type）（因为它对3和9有效）　　如果我们需要求出一个数的奇数位上的数字之和与偶数位上的数字之和的差值去检查这个差值能否被给定数整除，那么这个规则被称为11类型规则（11-type）（它对11有效）。　　有些情况下我们应当把除数分解成一些因数然后检查是否满足一些不同类型的规则（2-type,3-type,11-type）。例如，对于除数6我们需要检查2-type和3-type规则，对于除数66我们检查所有的三种类型的规则。这样混合的整除规则被称为6类型规则(6-type)。
最后，有些除数是所有类型的规则都无效的：既不是2-type，也不是3-type，也不是11-type，也不是6-type。这样的数最小的是7，所以我们称在这种情况下神秘的7类型规则(7-type)有效，也就是那种Vasya还没有发现的类型。　　Vasya的梦想是对所有可能的数求出整除规则。他将不仅仅停留在10进制下。由于有十分多的数字，他不能全都自己求。Vasya要求你去写一个程序求出在b进制下除数为d的整除规则类型。

算法讨论

一点意思都没有的题目，如果是3-type，那么$(b-1) \mod d = 0$，如果是2-type，那么$b \mod d = 0$，如果是11-type，那么$\text{lcm}(b-1, b+1) \mod d = 0$。

时间复杂度$O(1)$，代码很短。

10 Volume X

10.1 176E Archaeology

题目大意

现在你需要帮助一批在太平洋某个小岛上的研究人员，他们研究的是很多年前居住在这个岛上的古代部落文化。　　他们总共发现了N个村落，其中某几对村落之间有道路相连，这些道路从两个方向都可以走。总共有N-1条道路，并且任意两个村落能够互相到达。　　这些部落并不和平，经常会有战争。在一次战争结束后，某些村落将会被完全摧毁，在经过一段和平的年代后，某些村庄可能会被重建。　　在任意一个时刻，任意两个没有被摧毁的村庄之间的道路是被人们使用的。换句话说，找到一个最小的边集，它们把所有的未被摧毁的村庄连在一起，使它们两两可达。注意在这个岛的历史上，只存在这些研究者发现的N-1条道

路。　　研究人员认为观察不同时间使用道路的长度和能够帮助理解这个部落的文化，并解决一些历史问题。　　你会得到这个部落的所有历史。你的任务是在某些时刻，确定使用道路的长度和。

算法讨论

求出DFS序，我们需要知道每次操作的点和已经在集合内的所有点LCA最深的是哪个，既然LCA最深，说明在DFS序中离得很近。如此便可以用一个平衡术来维护已经加入了哪些标号和获得一个标号周围的DFS标号。

这样再加上求LCA，便可以动态维护所有的路径和了。

10.2 196D The Next Good String

题目大意

给一个仅由小写字母组成的字符串s和一个正整数m　　要求一个长度与s相同的仅由小写字母组成的字符串t，要求t的字典序大于s且不包含长度大于等于m的回文子串。

算法讨论

我们注意到不包含长度大于等于m的回文子串，就是不包含长度为m 和$m+1$的。

然后我们直接暴力DFS即可。

时间复杂度不好估计，实际速度很快。

10.3 198E Gripping Story

题目大意

在一个二维平面上，你现在的位置在(x,y)同时你手上有一块磁铁。　在这个平面上，还有N块散落的磁铁，每个磁铁都可以抽象成一个点，你的目标是吸引最多的散落的磁铁。
每一块磁铁都有五个属性x,y,m,p,r分别表示磁铁的横坐标，磁铁的纵坐标，磁铁的重量，磁铁的吸引力，磁铁的吸引半径。一块磁铁想要把另一块磁铁吸过来的条件，有两条。

1. 被吸引的磁铁和吸引的磁铁之间的距离小于等于吸引磁铁的吸引半径。这里距离计算的是欧几里得距离。

2. 被吸引的磁铁的重量小于等于吸引磁铁的吸引力。

任何被你吸过来的磁铁都可以用来吸引新的磁铁。每块磁铁可以吸引无数多次，但是每次只能有一块磁铁在吸引，不能多块同时吸引。同时你的位置(x,y)也是不变的。　　现在你想要知道，你最多可以吸引多少散落的磁铁。

算法讨论

本题大概相当于二维平面内有一些点，然后每个点表示一个矩形，我们要把这些点取走，被已取走或开始的矩形覆盖的点就可以被取走。问最多可以取多少。

直接树状数组套平衡树，暴力BFS即可。

时间复杂度$O(n \lg^2 n)$。

10.4 200E Tractor College

题目大意

给定c_3, c_4, c_5和m，求一组解w_3, w_4, w_5，使得$(0 \leqslant w_3 \leqslant w_4 \leqslant w_5)$，且$c_3w_3 + c_4w_4 + c_5w_5 = m$，且$\text{abs}(c_3w_3 - c_4w_4) + \text{abs}(c_4w_4 - c_5w_5)$最小。

算法讨论

直接暴力即可。注意多加剪枝和循环的顺序和方向。

时间复杂度不好估计，代码很短。

10.5 200A Cinema

题目大意

给你一个n行m列的零一矩阵A,每个元素初始值为0，再给你k个这样的操作:

给出(x_i, y_i)，你得按要求找出(a_i, b_i)，将矩阵A的点(a_i, b_i)赋值为1,并且输出(a_i, b_i)。要求$A(a_i, b_i) = 0$，且(a_i, b_i)与(x_i, y_i)，曼哈顿距离尽可能小，如果有多个满足条件的现要求a_i尽量小，再要求b_i尽量小。

算法讨论

直接暴力。记下每个点分配的曼哈顿距离最远是多少，然后和周围一定范围内的更新一下。

时间复杂度不好估计。

10.6 201E Thoroughly Bureaucratic Organization

题目大意

$\{a_n\}$是1到$n(n \leq 10^9)$的一个排列。每次可以至多询问$m(m \leq 10^9)$ 个位置上的数，询问的结果是立刻返回的，但是询问返回的结果是一个无序的集合，表示这m个位置上的数都有什么，问至少询问几次可以确定这个数组？本题多组数据，组数$t(t \leq 1000)$。

算法讨论

直接做显然是有困难的，我们设想一个神奇的函数$F(m, k)$表示每次询问最多m个，询问k次可以确定的数组的最大大小。如果我们能在可以接受的时间内计算$F(m, k)$，这个函数显然是关于k单调增加的，那么原问题便可以用二分最终答案得到解决。

我们先讨论给出一套n, m, k和询问方案，是否可以确定这个排列。不妨对于n个人，每个人对应一个长度为k的$0, 1$串。第i位为1表示在第i次询问中有提到这个人，如果为0则表示没有提到。

定理10.6.1. *当且仅当k个串两两不同，可以确定这个数组的。*

证明. 充分性：对于每个人，我们只需要找一个数出现且仅出现在包含他的询问中。这个数一定存在，因为这些询问都包含他；而且唯一，因为没有第二个人和他同时只在这些询问中有提及。

必要性：不妨设i和j的串一样，我们假设得到了一组解$\{a_n\}$，我们交换a_i,a_j，这时便有了一组新的符合条件的解。 □

我们同时注意到，对于每一位，我们需要让含有1的人数小于m。

定理10.6.2. *如果所有人都不一样且1的个数和小于km，那么便可以通过调整使得对于每一位，含有1的人数小于m。*

证明. 我们不妨假设第i位的1的个数超过了m，一定可以找到一位j满足1的个数小于m，然后我们假定有x 个人的字符串满足i位是1，j为是0。假定有y个人的字符串满足i位是0，j为是1，显然$x > y$。然后我们试图把这x个字符串的每一个第i为变成0，第j位变成1，看是否在那y个中出现过。一定存在一个没有出现的。然后照此进行修改，可使修改后的结果满足第i位1的个数减少，第j位1的个数增加且不超过m。不断照此修改，可以在有限步内修改成满足条件的。 □

这样我们就得到了一个简单的贪心算法，因为对于每个人1的个数尽可能少的优越性是显然的。

我们从1到k枚举1的个数，设t为当前剩余的1的个数，然后便可以得到$\min(C_k^i, \frac{t}{i})$个字符串含有i个1。最后只需要简单的求和即可。这样做时间复杂度大概是$O(\log n)$。

对于单组数据时间复杂度为$O(\log^2 n)$，总复杂度为$O(t \log^2 n)$。这是我的做法。

算法具体实现过程中要注意防止越界，组合数可能非常大。

10.7 201D Brand New Problem

题目大意

现在有n个单词组成的一句话s_0，以及m个由若干单词组成的长句子s_i，若s_0的某一个排列是长句子s_i的子序列，则称s_0同s_i相似，定义两者的差异度p为满足相似条件的排列的倒置数量（即在s_0的原排列与当前排列中相对位置发生改变的单词对数）的最小值，现在需要你求出在m个长句子中与s_0差异度最小的句子，若存在多个，则取编号最小的句子。

算法讨论

显然是动态规划。

我们定义$f_{i,j}$为出现了i的s_0（二进制表示），然后出现了j个倒置，的最短长度。然后处理出来从每个点之后第一次出现每个串是什么时候。

剩下的就是暴力了。

时间复杂度为$O(mn^2 2^n)$，代码很短。

10.8 204E Little Elephant and Strings

题目大意

给定n个字符串，询问每个字符串有多少子串（不包括空串）是所有n 个字符串中至少k个字符串的子串，总字符数$l(l \leqslant 10^6)$。

算法讨论

都连起来后缀数组。从前往后维护单调队列，保证这个区间内有k个字串。每次将这些区间更新这些后缀的最长合法前缀。最后统计答案即可。

时间复杂度$o(l \lg l)$

10.9 207B Military Trainings

题目大意

n个人排成一行，第i个人一个属性a_i，他们要从开始传递信号，从i传递到j需要1单位时间，还需要满足$i < j$且$i \geqslant j - a_x$，x为在j的人的编号，每传递一次最后一个人都要到最前面。

算法讨论

先倍长数组，然后直接倍增即可。算每个人2^i次最远能从哪里被传到。

至此本题做完，时间复杂度$O(n \lg^2 n)$，这是我的做法，代码很短。

10.10 207A Beaver's Calculator

题目大意

n位科学家，编号从1到n。第i位科学家给这个计算器带来了k_i个计算题。第i个科学家带来的问题编号1到n，并且它们必须按照编号一个一个计算，因为对于每个问题的计算都必须依赖前一个问题的计算结果。每个教授的每个问题都用一个数$a_{i,j}$来描述，i是科学家的编号，j是问题的编号，$a_{i,j}$表示解决这个问题所需资源单位的数量。这个计算器非常不凡。它一个接一个的解决问题。在一个问题解决后，并且在下一个问题被计算前，计算器分配或解放资源。计算器中最昂贵的操作是解放资源，解放远远慢于分配。所以对计算器而言，每一个接下来的问题所需的资源不少于前一个，是非常重要的。给你关于这些科学家所给问题的相关信息。你需要给这些问题安排一个顺序，使得“坏对”尽可能少。所谓“坏对”，就是相邻两

个问题中，后一个问题需求的资源比前一个问题少。别忘了，对于同一个科学家给出的问题，计算它们的相对顺序必须是固定的。输出方案。

算法讨论

虽然题目很长，但是实质就是每个人分开做，最后取较大值，分开做的时候便是直接贪心。整个问题毫无难度。

时间复杂度$O(nk)$，代码很短。

11 Volume XI

11.1 167D Wizards and Roads

题目大意

给定$n(n \leqslant 10^5)$个点，第i个点p_i为$(x_i, y_i)(x_i \leqslant 10^9+7, y_i \leqslant 10^9+7)$，点是随机的，保证没有任意两个点在同一水平线或者竖直线上。当两个点p_i, p_j，满足如下条件的时候他们之间可以连上一个边。

当且仅当任何在拐角中的点p_k均满足有一个点p_l不在拐角内。且x_l在x_i, x_j之间。

所谓拐角就是指满足下面两个条件之一的区域，不妨认为$y_i < y_j$。

- $\min(x_i, x_j) \leqslant x \leqslant \max(x_i, x_j)$且$t >= t_i$
- $y_i \leqslant y \leqslant y_j$且$(x - x_j) \times (x_i - x_j) \geqslant$

接下来有$m(m \leqslant 10^5)$个询问，对于每个询问，是一个区间$[L_i, R_i]$，问所有满足$L_i \leqslant x_i \leqslant R_i$的点$p_i$中，最多能取出多少对，使得每对之间有边。

算法讨论

我们无视那么多的条件，我们让所有区域内都没有点就是了。

我们对这些点建立一个神奇的二叉树，满足中序遍历之后得到的序列x_i 是单调增加的，而每个点的父亲的y值均比自己的大。因为这个数据是随机的，所以树高比较小。（应该是$O(\lg n)$级别的？我不知道。）

然后我们注意到，每个点都可以和他的父亲连上一个边都是合法的，而且每次询问的区间都是由很少的子树组成的。我们只需要事先处理出来这些子树的结果，然后询问的时候递归往下找即可，中间转移类似树形动态规划。

这个的时间复杂度是$O((n+m)\lg n)$，代码很短。

11.2 209C Trails and Glades

题目大意

给一个$n(n \leqslant 10^6)$个点，$m(m \leqslant 10^6)$个边的无向图（可以有重边和自环）。要求加尽量少的边，使得存在一条从1号点出发，并回到1号点，且经过所有边的回路。求至少加多少个

边。

算法讨论

我们假象加入一条1到1的边，这样我们就只需要找一条经过所有边的回路即可。

接下来我们要满足的条件可以总结为两点，使这个图联通，使每个点度数为偶数，下面我们把度数为奇数的点称为奇点，度数为偶数的点称为偶点。

第二个条件比较好满足，只需要统计出来奇点的个数c即可，我们需要$\frac{c}{2}$个边保证没有奇点。

第一个条件，显然和联通块有关系，我们用并查集维护出所有联通块，并统计出没有奇点的联通块且这个联通块有边的个数d，我们给出证明。

证明. 如果两个联通块内均有奇点，那么让他们联通我们不需要加边，只需要通过调整即可。因为对于消除奇点来说，边另一端连在哪里不重要。如果两个联通块中有一个有奇点，那么让他们联通我们需要加一条边。如果两个联通块中都没有几点，我们需要加两个边。可以认识到，加的边的个数即是d。 □

最终答案就是$\frac{c}{2}+d$，但是要注意如果只有一个联通块，且这个块内没有奇数度的点要输出0。

至此，本题做完，时间复杂度$O(n+m)$，代码非常短。

11.3 212B Polycarpus is Looking for Good Substrings

题目大意

现在有一个的字符串，都是有小写字母构成。再给你很多个集合。对于每个集合，求以下子串$s[a,b]$的数量，字符串长度$l(l \leqslant 10^6)$：

1. 子串中出现的字母必须出现在集合中。

2. 集合中的字母必须在子串中找得到。

3. 不存在比这个子串更长的子串$s[x,y]$,使得$s[x,y]$满足1,2，且$(x \leqslant a \leqslant b \leqslant y, y-x+1 > b-a+1)$

算法讨论

我们离线处理所有询问，我们用一个二进制来存储询问，并将询问排序，方便通过二分索引。（当然如果不在乎效率可以直接用STL map。）

我们考虑从每个点开始的串，从前往后扫，如果这个字母在当前的状态中已经出现过，便直接执行下一次字符，然后我们更新答案（因为前后这个字母都没有出现过这个字母一定会被计数），然后我们看看下一个字母是不是和第一个字母之前的字母一样，如果一样就要直接诶退出了，否则便更新当前状态，并继续执行下一个字母。

算法复杂度是$O(n \lg m)$，代码比较短。

11.4 212D Cutting a Fence

题目大意

给一个长度为$n(n \leqslant 10^6)$的数组$a_i(a_i \leqslant)$，有$m(\leqslant 10^6)$ 个询问每次询问一个$k(k \leqslant n)$，求长度为k的区间的期望最小值是多少。

算法讨论

显然是先预处理出来n个结果，然后我们$O(1)$的询问。

预处理的时候，我们考虑最小值的位置（如果一个区间有多个最小值，我们考虑最靠左的最小值的那个位置。）于是我们对于每个位置i，都得到了一个L_i和R_i表示如果区间$[l,r]$满足$i-L_i \leqslant l \leqslant r \leqslant i+R_i$，那么这个区间的最小值为$a_i$，这个$L_i$和$R_i$的求法可以使用单调队列，然后我们只需要考虑我们知道了L_i, R_i，即可。

我们注意到，求一个二阶差分之后，只修改了四个地方，然后我们便可以直接修改二阶差分，然后求两次前缀和即可。

算法复杂度是$O(n+m)$，代码比较短。

11.5 212C Cowboys

题目大意

一个环形字符串（但是起点不一样是不同的字符串），长度$l(l \leqslant 100)$ 我们考虑所有AB这个字串，我们将每个AB同时且瞬间变成BA，得到一个新字符串，现在我们给出这个字符串，问原先的字符串是什么。

算法讨论

很简单的动态规划。由于是环形的，我们考虑两次DP，第一次是假定原先的第一个字符是A，第二次假定原先的第一个字符是B，然后A_i就是原串最后一个是A的方案数，B_i就是原串最后一个是B的方案数。做两次初值不一样转移一样的动态规划，然后将答案相加即可。

至此本题做完，时间复杂度$O(n^2)$，这是我的做法，代码很短。

11.6 213E Two Permutations

题目大意

长为$n(n \leqslant 200000)$的排列a 和长为$m(m \leqslant 200000)$的排列b 。求有多少不同的整数d满足，存在序列$c(c_1 = a_1+_d, c_2 = a_2+d, \cdots, c_n = a_n+d)$ 是b的子序列。

算法讨论

我们考虑数列$pa(pa_{a_i} = i)$，和$pb(pb_{b_i} = i)$。我们注意到原题就是求pb中有多少个子串的相对大小情况和pa一样。这是一个经典问题了。可以用kmp算法解决。

至此本题做完，时间复杂度$O(n+m)$，这是我的做法，代码很短。

11.7 217C Formurosa

题目大意

你有$n(n \leqslant 10^6)$个布尔变量，保证不都一样，这一有一个表达式F，你可以把变量代入进去，然后你会得到这个表达式的结果，你可以代入任意次，求你是否可以得到这n个布尔变量的值。

算法讨论

首先一个显而易见的结论，如果存在一个输入s，将s中的每个都取非，得到s'。如果$F(s) \neq F(s')$。那么便可以得到这n个变量的值。

我们考虑对于每个表达式是不是有这三个性质，存在一种输入s

- 使得$F(s) = F(s') = 1$
- 使得$F(s) = F(s') = 0$
- 使得$F(s)! = F(s')$

如此我们只需要递归的维护每个表达式的这三个性质即可。

时间复杂度为$O(n)$，代码很短。

11.8 229E Gifts

题目大意

有n组物品，第i组有p_i个物品，第j个价值为$p_{i,j}$。要从中选m个，会从其中可能得到最大的m个的中选取任意一种方法。但是如果我们在第i组中要了q_i个，会随机返回q_i个而不一定是最大的。求都是最大的概率。$(n, m(n, m \leqslant 1000), c_{i,j} \leqslant 10^9)$。

算法讨论

注意到每组之间都是独立的。然后我们只需要决定最后那些等价值的次数应该在哪些组里面取。

整个问题变成了a个的数，取b个，求期望乘积。

这个显然可以直接动态规划做。

时间复杂度$O(nm)$，代码很短。

12 Volume XII

12.1 75E Ship's Shortest Path

题目大意

给定一个起点，一个终点和一个凸多边形（点数$n(n \leqslant 30)$），保证起点终点不在多边形内，坐标范围$(x, y)(-100 \leqslant x \leqslant 100, -100 \leqslant y \leqslant 100)$。你只能在这样的位置上移动，在起点和终点之前或者是凸多边形的边缘，每走一单位长度需要一单位时间。也可以在多边形内移动，在多边形内每走一单位长度需要二单位时间，求从起点到终点的最少时间？

算法讨论

我们注意到最优解一定是下面两个之中的一个。

1. 直接线段过去，这种情况只需要求线段和凸多边形的相交区域。然后直接计算即可。

2. 走线段直到与凸多边形相交，然后从某一边绕道而行。

这显然是正确的。

这个的时间复杂度是$O(n)$，代码很短。

12.2 113D Museum

题目大意

有$n(n \leqslant 22)$个房间，有两个人刚开始在的位置分别在x, y，他们想相遇，第i个房间有一个概率p_i，这个概率表示如果这个人在这个房间内，那么下一分钟有p_i 个概率留在这个房间，有$1 - p_i$ 概率去和他相邻的其他房间，去这些房间是等概率的。问最终在每个房间相遇的概率。（相遇之后就不会在移动了）

算法讨论

我们把一个元组(i, j)定义为一个状态，这样一共有n^2个状态，我们考虑在这些状态中转移，最后要到的状态是(i, i)，我们定义$p_{(i,j)}$为到(i, j)这个状态的期望次数，显然由于到(i, i)之后就不再转移了，所以期望次数即是概率。然后我们列一个方程做消元即可。

这个的时间复杂度是$O(n^6)$，代码很短。