结果提交类问题

重庆外国语学校 雷环中

【关键词】

结果提交 数据分析 随机化 策略

【摘要】

结果提交类问题的历史非常短，但其崭新的模式、效率观、时空观，以及独特的解题技巧、解题策略，和对各种方法的鼓励，对选手更加深入、更加全面的考察而很快受到人们的青睐，在短期内快速的传播并发展，以至成为当今信息学奥林匹克中的一种重要题型。本文就结果提交类问题的两种重要技巧——数据分析和随机化展开讨论，提出此类问题所需要的不同策略。用一句话概括本文的核心，即：在最短的时间内得到正确结果，而不是过程。

【目录】

【正文】

一、引言

结果提交类问题可算是当今信息学奥林匹克竞赛中最年轻的一类问题，然而 IOI 连续两年出现此类问题的事实，使其成为主流竞赛中不争的必考题。其中最主要的原因，还是由于此类问题灵活新颖，以及命题人对殊途同归的倡导和对新奇方法的鼓励。

纵观在近两年来主流竞赛中出现的 Crack the code，Double Crypt，Birthday party，Tetris，Xor 等五道题目，可以说道道精彩，其中不仅充满了巧妙的构思，解题的方法也是百花齐放。但如何才能正确、迅速地完成此类问题，还需要我们深入地思考。本文以上述五道题目为例，分析结果提交类问题的一些特点，介绍一些针对结果提交类问题的特殊方法和技巧。

伴随着结果提交问题的出现，很多方面都发生了深刻的变革：

1. 考察内容的变革

在信息学奥林匹克竞赛发展已较为成熟的今天，仅仅就选手算法和数据结构方面能力作为主要考核标准是远远不够的。结果提交类问题的出现，使得考察的核心还涉及了一些更深层次的东西，如选手的思维能力、创造能力，全局策略时间策略等等。这些方面的考察在以往题目中同样存在，但似乎都没有结构提交类问题体现得这样充分，考察得如此精妙。

2．区分度的变革

一道题目对选手的区分度发生了深刻的变革。我们不能仅仅依靠对算法时空复杂度的分析来衡量算法的优劣，还应该综合考虑，衡量我们完成输出文件所花的时间。原因很简单，在以往，1 秒钟出解的程序和 10 秒钟出解的程序，测试的结果一般是前者得分要高不少；而结果提交类问题如果如果结果均正确，一名选手花 20 分钟，另一名花时 2 小时，前者则明显胜出，不论二人使用何种方法。

3．题目的变革

我们后面可以看到，如 Birthday Party 这样的题目，是否结果提交，完全决定了我们的思考方式，决定了算法。既然命题者将一道题目以结果提交的形式出现，就说明并不希望选手将其视为一道普通题目，而是希望选手大胆使用各种方法，以在最快时间内得到正确结果。倘我们仍以老的思路来看待此类问题，不仅违背了命题者的意愿，浪费了所提供的输入文件，更重要的，我们将浪费更多的时间，甚至得不到理想的方法。结果提交，不是让我们放弃对完美算法的追求，而是鼓励我们站在综合效益的角度去得到结果。

4．评卷方式的变革

大家都清楚，信息学奥林匹克的评卷是黑箱的，然而以前的黑箱似乎显得相对较狭义。自从结果提交类问题出现以来，黑箱的涵义得到了进一步拓展，更突出了对结果的唯一追求。过程，或者说是程序，更加显得无足轻重了。

我们用一句话来概括解决结果提交类问题的原则就是：永远提醒自己，你所需要的只是在最短的时间内得到正确结果，而不是过程。下面的讨论，也都是围绕此原则展开的。

二、数据分析

数据分析的方法始于结果提交类问题，由于这种方法对选手的观察、思考、猜想及动手实践能力的要求较其他的方法更高，而且灵活多变，所以备受重视。下面以 Crack the code 和 Birthday party 为例进行说明。

[例一]Crack the code(Baltic OI 2001)

题目大意：

有五个文件 cra*.out，与其对应的，有 cra*.txt，已知每一对文件出自同一篇文章。对于每个.out 文件，有十个未知的 Byte 型数 $a_1 \ldots a_{10}$，我们对文件的第 i 个字符 c_i 进行下述变换：

1．　如果 c_i 不是英文字母，则不做变化；

2．　如果 c_i 为小写字母，则变为大写，转 3；

3．如果 c_i 为大写字母，则对其连续进行 $a_{(i-1) \bmod 10+1}$ 次取后继操作。我们视‘A’的后继为’B’，‘B’的后继为’C’…… ‘Z’的后继为’A’。

然后将 c_i 依次写入对应的 cra.in 文件。你的任务是根据提供的 cra*.in 和 cra*.txt(都在 11KB 以内)，得出 cra*.out。

分析：

这道题是没有对于任何数据都非常有效的算法的，因为它毕竟是一种比较成形的加密算法。这也是命题者将此题作为结果提交问题的根本原因，只需你要破解给出的数据即可。

本题的解决方法很多，后面我们将要提到。还是先让我们试着用数据分析的方法来考虑此题，来看看数据分析方法的威力。

首先我们注意到本题数据只有 5 组，这和近年来 10 组甚至更多的测试数据不大一致，或许是在暗示这道题可以手算，换句话说，就为数据分析的方法提供了可能性。

注意到题目中的五篇文章都是英文的。我们清楚，英语中的许多高频词汇，如 a，I，the，he，are，of，in，after 等都是很短的，同时，单词长度越短，可能的情况就越少，如果长度为 1，我们就基本可以肯定是 a 或 I 了。

问题的关键是求出 $a_1 \ldots a_{10}$，之后，cra.out 自然也就很容易得到了。而 $a_1 \ldots a_{10}$ 反映在微观上，就是原文和加密后的文章中每个单词（当然更是某些单词）的对应字母之间的差异。也就是说，我们只要确定了少数原单词和加密后单词的一一对应关系，$a_1 \ldots a_{10}$ 也就浮出水面了。比如 $A_1A_2 \ldots\ldots A_k$ 是原单词，$B_1B_2 \ldots\ldots B_k$ 是加密后的单词，A_1 是文章的第 m 个字母，则我们可以得到 $A_j \xrightarrow{\text{进行a}_{(m+j-2) \bmod 10+1}\text{次取后继操作}} B_j$，其中 j=1，2，…k。反之，自然有 $B_j \xrightarrow{\text{进行a}_{(m+j-2) \bmod 10+1}\text{次取前趋操作}} A_j$。

基于上述思路，我们完全可以仅用猜想+试探少数单词间对应关系的方法处理本题，当然还需要编两三个十余行的辅助小程序来提高效率。这种方法全靠观察能力和猜想能力，不涉及算法的复杂度。

下面仅以数据一和数据五为例进行简单说明。

数据一：

cra1.txt：

ALFRED TARSKI WAS ONE OF THE GREATEST MATHEMATICIANS, LOGICIANS
AND PHILOSOPHERS OF THE PASSING CENTURY, AND ONE OF THE GREATEST LOGICIANS
OF ALL TIME. 后略。

cra1.in

JK XHLMGXC PP ZEU OWL BM 1939 VP ZEU CRGBSHVOLD IJ UHGU XUK DYYXTM HPJ
IYPIO MGLTK BLYV DBMJG, GCVCPTTSLF LFHMX LM SOG NXHPECW TUKBBHMMER ZUF ZEUH,
EQMDY 1943, CZ QXY YYBULTYFJS SQ VZSKLLHHML TS IGXHUFIJ. 后略。

对于输入文件，稍加观察我们便可以发现，位于第三行的 EQMDY 1943 是突破口。英语常识告诉我们，EQMDY 只能是 AFTER！同时，EQMDY 的'E'是文章的第 116 个字母，同时有 $AFTER \xrightarrow{(4,11,19,25,7)} EQMDY$，这样，10 个数就被破了 5 个，具体地说，就是 $a_6 = 4$，$a_7 = 11$，$a_8 = 19$，$a_9 = 25$，$a_{10} = 7$。我们用得到的这 5 个数还原文本之后，就可以得到：

JK XHLIVED IP ZEU OSA IN 1939 OP ZEU CNVITAVOLD IF JOHN XUK DYUMAN APJ
IYPED THETK BLYR SINCG, GCVCLIATEF LFHMT AT THG NXHPARD UNKBBHMITY ANF ZEUH,
AFTER 1943, CZ QXY UNIVETYFJS OF CALKLLHHIA AT BGXHUFEY. 后略。

很多单词都已露出了"马脚"，选择一两个较长的单词，即可轻松迅速地破解全文。如第二行的 THG NXHPARD UNKBBHMITY，我们不用想就知道应该是 THE HARVARD UNIVERSITY；还有第三行的 UNIVETYFJS OF CALKLLHHIA，我们也很容易看出是 UNIVERSITY OF CALIFORNIA，因为 cra1.txt 告诉我们这段文字是在描述一个人的生平。之后再做一次转换，得到 a_1，a_2，a_3，a_4，a_5，于是就可破解全文了。

数据五：

cra5.txt：

THE PLURAL IS USED HERE BECAUSE THE TERM "HAHN-BANACH THEOREM"
IS CUSTOMARILY APPLIED TO SEVERAL CLOSELY RELATED RESULTS.

cra5.in

YSOEGOCYTV: OVN ZNGU IUIXS XB T ZXFTTJURYTR EUHBJF BLTIN N
NA OVN RXICEW AKOLA Q* CQEXM ZZNIXTCI FZZ HQA VUWJNVPCDO
EOWUFZ AIWYMOXDFTN CW T.

GUCU YPVH JZWOCYTV VBM OVGUQW UPZCEIRRSFBDCW WKK MUKQISM EG D* KO (...)
NB DG LHXGA JMIO HQALK XFJZVHRKGY ME NVYSNZ FGTU C* QIHX W OKLJTZ NDJYX.

对于官方的参考程序而言，数据五是最难的，但对于我们数据分析的方法而言，却是最简单的，我们甚至可在一分钟内将它破解。

根据 cra5.txt 文件，我们可以大致判断这是一篇理科方面的文章，从英语语法的角度我们知道，cra5.in 开始的 OVN 可能是 the。"YSOEGOCYTV:"似乎是引出什么东西的定义。YSOEGOCYTV 共有 10 个字母，我们可以立即想到或许是 DEFINITION 这个单词。当然这不一定是对的，但试验之后，我们发现，文章到

此已被完全破解:

DEFINITION: THE DUAL SPACE OF A TOPOLOGICAL VECTOR SPACE X
IS THE VECTOR SPACE X* WHOSE ELEMENTS ARE THE CONTINUOUS
LINEAR FUNCTIONALS ON X.

NOTE THAT ADDITION AND SCALAR MULTIPLICATION ARE DEFINED IN X* BY (...)
IT IS CLEAR THAT THESE OPERATIONS DO INDEED MAKE X* INTO A VECTOR SPACE.

正如我们所看到的，人脑在此处起了决定性的作用。如果说以前的题目是透过程序来对选手进行考察的话，那这种题目就是直接与选手面对面的对话。

下面我们来考虑如何通过程序来完成该问题。

由于加密文本是采用英语这种仅由 26 个字母组成的语言，而且理所当然，每种字母在一定文段中出现的频率是不相同的。如'S'出现的频率一般都比'V'高得多(我们统一使用大写字母)。基于此思想，我们可以设计一种频率比较的方法。

用 $(P_A,\ P_B,\ P_C,\ldots\ldots P_Z)$ 来表示给出的未加密文本 cra.txt 中每个字母出现的相对频率，也就是

$$P_{ch}=\frac{\text{文本中字母}ch\text{出现的次数}}{\text{文本中字母的总数}}(ch='A','B',\ldots\ldots'Z')$$

我们分别计算 $a_1 \ldots a_{10}$。对于 $a_i(i=1,2,\ldots\ldots 10)$，先将位于加密文件中第 i，$i+10$，$i+20\ldots\ldots$位的字母做成集合 S，然后对于 S 做类似的统计：我们用 $(Q_A,\ Q_B,\ Q_C,\ldots\ldots Q_Z)$ 来表示 S 中每个字母出现的相对频率，即

$$Q_{ch}=\frac{S\text{中字母}ch\text{的数目}}{S\text{中字母的总数}}(ch='A','B',\ldots\ldots'Z')$$ 。

之后从 0 到 25 枚举 a_i，同时作变换 $Q_{ch}\xrightarrow{\text{字母}ch\text{取}a_i\text{次前趋得到字母}ch'}H_{ch'}$，最后比较 P 和 H 的相似程度，取最相似时的值作为真正的 a_i。令 $f(P,H)$ 表示 P 和 H 的相似程度，于是问题的关键就转化为如何设计这个估价函数。一方面为了更好地反映各种函数之间的差异，另一方面由于 $a_1 \ldots a_{10}$ 间没有必然的联系，所以我们后面按照密码的个数来说明，一个测试数据有 10 个密码。

方法一：最自然的一种，令 $f(P,H)=\sum_{ch='A'}^{'Z'}|P_{ch}-H_{ch}|$；

方法二：上面一种似乎不能很好地反映其相似程度，特别是在相差很微妙的时候。于是我们可以稍加改进：令 $f(P,H)=\sum_{ch='A'}^{'Z'}(P_{ch}-H_{ch})^2$；

方法三：第二种方法却又显得将权重夸大了，我们可以进一步改进：令 $f(P,H)=\sum_{ch='A'}^{'Z'}\frac{(P_{ch}-H_{ch})^2}{P_{ch}}$。注意到这里的 P_{ch} 不能太小(比如不能小于0.001)，于是我们将函数修正为 $f(P,H)=\sum_{ch='A'}^{'Z'}\frac{(P_{ch}-H_{ch})^2}{P_{ch}+Delta}$，其中 Delta 等于最小的一个非零 P_{ch}。

下面是三种方法的比较，表中的数字为每个数据算对的密码个数：

	数据一	数据二	数据三	数据四	数据五
方法一	10	10	10	10	8
方法二	10	10	10	10	8
方法三	10	10	10	10	10

我们发现，随着输入文件的长度的缩短，样本容量越来越小，这是统计方法的大忌，前面两种方法便在数据五这种很小的样本容量面前失败了。

为了更好的区分程序的性能，我们另外设计四个数据，这四个数据都是相当短的，难度比前面五个大得多。

cra6.txt

It passes all vertical lines in increasing order of their x -values and keeps a
list of active y -intervals. At each "stop", i.e. for every vertical line,
it does the following:

cra6.in

NWOXNVSG CRF Z -GMXZHVLO UQ WLJ VYMESPWV WYUW
ERDI VKI XOGM XP UJH MSZLZEKM VKEY OZ BQO VPLSS UM IUV BEWMAK F -QWDFTYEQY,

cra6.out

MULTIPLY THE X -DISTANCE TO THE PREVIOUS STOP
WITH THE SIZE OF THE INTERVAL THAT IS THE UNION OF ALL ACTIVE Y -INTERVALS,

cra7.txt

Scan-line solution, (modification of previous solution with added y-axis compression).

cra7.in

TEDR-QOUM BYMWWMTT DQCR MCCC NSWTNWFPWEYOVV. CRF UFES RPVN ST CUVFE VDNB SCQKJ 0... 30000 .

cra7.out

SCAN-LINE SOLUTION WITH LAZY IMPLEMENTATION. THE SCAN LINE IS ARRAY OVER RANGE 0... 30000 .

cra8.txt

My name is Mary.

cra8.in

J CP E HRLDNB HKUP. N GT VXD CCG.

cra8.out

I AM A CLEVER GIRL. I AM NOT BAD.

cra9.txt

All tests were generated automatically.

cra9.in

NQVX FXL ZJXEQP.

cra9.out

MOST ARE RANDOM.

下面是比较的结果。这次我们将官方程序及数据分析法一起加入比较。

	一	二	三	四	五	六	七	八	九	总计(上限 90)
方法一	10	10	10	10	8	8	5	1	1	63
方法二	10	10	10	10	8	8	3	1	1	61
方法三	10	10	10	10	10	7	5	2	1	65
官方程序	10	10	10	10	10	6	4	0	1	61
数据分析法	10	10	10	10	10	10	10	10	0	80

我们可以看到，由于问题本身的特殊性，没有一种方法可以给出非常理想的解答。其中四种程序实现的方法差异不大，随着样本容量的减小，正确率显著降低，直到数据八九的时候，可以说已经完全无能为力了。而数据分析方法一直表现良好，除非如数据九，文件内容简单得只有“NQVX FXL ZJXEQP.”，已经无从逻辑分析了，只有枚举。

另外值得一提的是，如方法一在数据五六的时候，得出的结果已经“比较”正确，这时候再结合数据分析的方法，就轻而易举了。这也说明，数据分析进行检查也是前面四种程序方法结束后的必经之路。

类似本题这样手工处理数据的方法在结果提交类题目中是非常普遍的，甚至可以说，用这种方法还能在相对更短的时间内得到正确答案(如本题)。既然如此，在比赛中又何乐而不为之呢？在不忽视“正统”方法的同时，我们应该针对题目的特点，想出更好的算法。

处理本问题的方法难以尽数，各种方法的优劣也难以仔细分辨，当然也包括数据分析方法。我想这也是命题人将本题作为结果提交类问题出现的原因之一

下面我们将这种数据分析的方法和程序方法进行比较：

	结果正确性	完成题目所需时间	易于实现性	对于加大测试数据数目的适应性
数据分析	好	短	很易	不好
程序方法	较好	较长	一般	好

结果是一目了然的。仅对于本题结果提交的特点而言，数据分析方法相对而言是最好的，但是如果测试数据是二十个，那数据分析就显得太慢了。字母频率

统计方法对于本题显得有些杀鸡用牛刀，而且结果的正确性不能得到很好的保证。但不论何种方法，都是需要人工+程序的，只是程度不同。

综上所述，不同方法的适应性是不同的，但在结果提交问题中，快速准确出解，才是最重要的。

[例二]Birthday party(CEOI 2002)

题目大意：

有 N 个人，另给出 M 条“要求”。

要求的定义如下：

1. *name*是一个要求。*name*是一个长度不超过20的字符串，表示一个人的名字。这个要求当且仅当*name*被选中时满足。
2. *-name*是一个要求。这个要求当且仅当*name*不被选中时满足。
3. 如果 R_1，……，R_k 都是要求，那么 $(R_1\&\ldots\ldots\&R_k)$ 也是一个要求。这个要求满足当且仅当 R_1，……，R_k 都被满足。
4. 如果 R_1，……，R_k 都是要求，那么 $(R_1 \mid \ldots\ldots \mid R_k)$ 也是一个要求。这个要求满足当且仅当 R_1，……，R_k 中至少有一个被满足。
5. 如果 R_1，R_2 是要求，那么 $(R_1 => R_2)$ 也是一个要求。这个要求不满足当且仅当 R_1 满足而 R_2 不满足。

提供给你 10 个输入文件 party*.in，每个输入文件包含 N 个人的名字和 M 个要求。对于每个输入文件，分别找出应选中哪些人，使得 M 要求全都被满足。(至少有一个解，你也只需要输出一个解)

从输入文件我们可以看到，$N \le 25000$，$M \le 129998$。

分析：

看完题目，我们仅从数据规模就可以感觉到这道题应该是线性规模或接近线性规模的，但此题的模型 SAT 却早被证明是 NPC 的，这种怀疑使我们有理由相信，此题的关键不是题目，而是数据。命题者对数据精妙的设计，也使本题成为结果提交类题目中最经典的一道。

每一个要求都是一个逻辑表达式，其中，我们视“-”为 NOT，“|”为 OR，“&”为 AND，“=>”为 THEN，将每个人名(即逻辑变量)赋为真或假，那题目也就是要找出一种赋值方案，使得所有的逻辑表达式值都为真。

下面我们来对数据进行一一分析：

Party1.in

5

adam

bill
cindy
don
eve
7
(adam | bill | cindy | don)
(-adam | -bill | -cindy | -don)
(-don | -eve)
(-bill | -bill | -bill)
(bill | adam)
(-adam | don)
(eve | cindy | bill)

这是最简单的数据，我们既可手算，又可通过更强大的程序完成。究竟选择哪一种，则是一个策略问题。如果后面有些更大的数据有着相同的性质，那我们应该采用程序完成。否则手算是最划算的，因为花一分钟动笔比设计一个上百行的算法要容易得多。

Party2.in
15
b
c
d
e
f
g
h
i
j
ba
bb
bc
bd
be
bf
100
(-g | -bb | -c)
(e | -be | -b)
(-i | -bd | be)
后略，形式同上面三行。

毫无疑问，枚举或者是任何更好的方法都可以对付这个数据，枚举的时间复杂度为$O(2^N \times M)$。

通过观察，我们还可以发现，如果将 *a*，*b*，*c*……*j* 分别和 0，1，2…9 一一对应，那么这些名字就是从 1 开始的 *N* 个自然数！后面的数据也满足这个特征，这为我们的数据处理提供了很大方便。这一点虽然很容易看出，但以前在题目中是不容易设计这种对的观察能力的考察的。也即，结果提交类问题，不仅给选手更广阔的空间，也将同样广阔的空间给了命题者。

Party3.in
35
b
c
d
其他名字略，规律如上。
700
(ba | -cb | -j)
(-cb | ch | bb)
(-ba | j | e)
后略，形式同上面三行。

形式和数据二是一样的，不过规模要大不少，如按 $2^{35}\times 700$ 估算，枚举算法已不能在较短时间内出解(事实上已远远超过 5 个小时)。不过一个很一般的基于“要求”的搜索算法就可以对付这个数据。进一步，我们在搜索中应该去掉那些到目前为止已经成立的要求，并且将那些除了某个变量为真该要求就不能成立的首先考察。

Party4.in
和数据二差不多，枚举或者是任何更好的方法都可以对付这个数据。不过值得注意的是，有下面六条要求的形式有所不同：
((bg & i) => (-be | -b | ba | j)
((-i | g | -bf | g | -b | -g) & (-bh | -b | -bd | -bb | bc | -ba))
(b => (-b => ((c | (e & f)) => ((g & -h) | (e | h)))))
((-bd & h & -be & bh & g) => h)
((-j & -bb & -bf) => (-d | -bb | -e))
(bd => (-bc => (g => (bb => (c => -be)))))

我们可以先用一般算法得出解，然后人工判断其是否符合这 6 个要求。要知道，仅为这 6 个要求而去编写大量代码是很不划算的。当然，这也是结果提交类问题所带来的好处。

Party5.in
N=500，*M* =5000
要求的形式：

全部为(N_1 => N_2)　(其中 N_1，N_2 为 *name* 或*-name*，下同)

Party6.in
N=1000，M =10094
要求的形式：

全部为(N_1 => N_2)

Party7.in
N=5000，M = 50000
要求的形式：

全部为(N_1 => N_2)

Party8.in
N=10000，M = 55000
要求的形式：

全部为(N_1 | N_2)

Party9.in
N=25000，M = 129998
要求的形式：

基本上为(N_1 | N_2)，另有六行(N_1 => N_2)

此外还有两行非常怪异的
(eebi => (eebf & -eead & -eeaj))
(bfjfg | fchf | fchf)

从数据四和数据九我们可以看出，如果不仔细观察或者是编写一个小程序来扫描，有些信息我们是很容易忽略的，这也是结果提交类问题的特点，以前完全程序处理，程序是不会疏忽的，但人就不同了。结果提交，带给我们的既是蛋糕，又是陷阱。

Party10.in
N=1000，M = 87543
这可以说是最可怕的一个数据，形式没有任何规律！但仔细看一看 M 个要求的尾部，我们发现：
(e => -e)
(-f => f)
(-g => g)
……
(jji => -jji)
(jjj => -jjj)
(baaa => -baaa)

似乎 1000 个名字都齐了，但唯独缺少 *b*，*c*，*d*！可我们再到回去看看最前面的三行要求：

(d => (-b & -c))
(b => d)
((-b => -c) & (-c => d))

这些情况已经足以出解了，因为由(name=>-name)我们可以得到不选 name。再结合余下的三行，我们可以得到唯一的解。根据题目所说的至少有一个解，并且只需要输出一个解，那么剩下的八万余行就可以完全不管了。

综合考虑：

数据一：　手算。因为后面数据的组织方式和这个数据完全不同。

数据二三四：　统一用一个搜索算法完成。事实上，从比赛策略的角度而言，我们甚至可以舍弃数据三，编一个枚举算法，这样可省些时间。

数据五六七八九：形式全部为(N_1 => N_2)或(N_1 | N_2)，需要一个线性规模或接近线性规模的算法。

数据十：　还是手算，不过这里的手算和数据一有点不同，实质上是手工+简单程序计算。

另外，对于数据四和数据九的几行特例，最好的方法是人工验证，简单高效。正所谓“具体问题具体分析”，我们根据数据各自的情况，将一个 NPC 的问题完全分解了。

至此，本题的核心已非常明显，即是数据五六七八九的 **2-SAT**。该命题和 POI 0106 Peaceful Commission，Baltic OI 2001 的备用题目 Excursion 相似。让我们先来看看这两道题目的模型：

Peaceful Commission(POI VIII Stage 2 Problem 2)

给出 *2N* 个点(*1≤N≤8000*)，将点 *i* 和 *-i*(*1≤i≤N*)作为一组，另外给出 *M*(*0≤M≤20000*)对整数 *i*、*j*，表示点 *i* 与点 *j* 不相容(*-N≤i*，*j≤N*，$i \neq 0$，$j \neq 0$，$i \neq j$)。求出一含 *N* 个点的集合 *S*，包括且仅包括每一组中的一个点，且 *E* 中的点均相容。

Excursion(Baltic OI 2001 备用题)

给出 *2N* 个点(*1≤N≤8000*)，将点 *i* 和 *-i*(*1≤i≤N*)作为一组，另外给出 *M*(*0≤M≤20000*)个整数对(*i*、*j*)(*-N≤i*，*j≤N*，$i \neq 0$，$j \neq 0$)。求出一含 *N* 个点的集合 *S*，包括且仅包括每一组中的一个点，且对任意一个前面给出的整数对(*i*、*j*)，点 *i* 和点 *j* 中至少有一个属于集合 *S*。

我们暂且不具体讨论这两道题，但我们发现，两道题的数据规模竟然都是一样的。事实上，后面我们可以证明，这两道题与本题都是本质相同的。

Party 的模型如下：

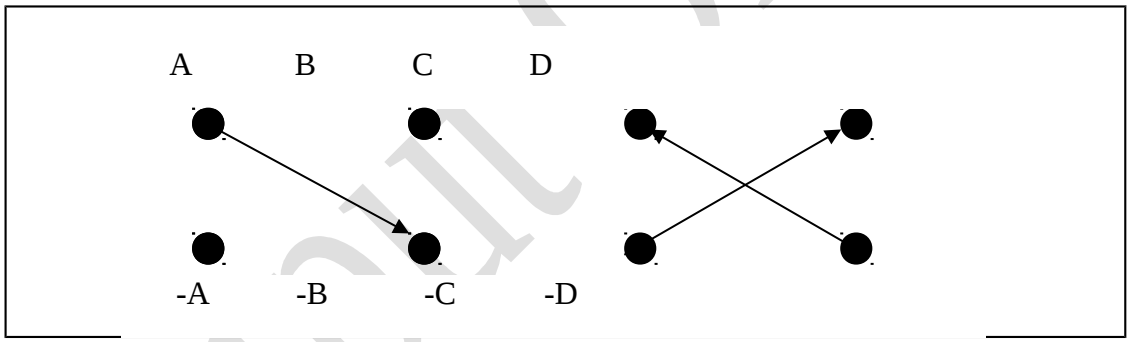
给出 *2N* 个点(*1≤N≤25000*)，将点 *i* 和*-i*(*1≤i≤N*) 作为一组，另外给出 *M*(*0≤M≤129998*)个要求，要求分两种，每个要求包括一个有序整数对(*i*、*j*)(*-N≤i*，*j≤N*，$i \neq 0$，$j \neq 0$)。求出一含 *N* 个点的集合 *S*，包括且仅包括每一组中的一个点，且满足所有要求。包括整数对(*i*、*j*)的要求被满足当且仅当：

1. 如果该要求是第一类要求，则如果点 $i \in S$，必有点 $j \in S$。
2. 如果该要求是第一类要求，则点 *i* 和点 *j* 中至少有一个属于集合 *S*。

我们先分析 Party：

首先，根据定义有-(-name)=name。对于每个名字 *name*，我们建立顶点 V_{name} 和 V_{-name} (由于名字和整数是一一对应的，后面我们将不再区分 V_{name} 和 V_i)，V_{name} 表示要选，V_{-name} 表示不选。V_{name} 和 V_{-name} 也分别代表一个布尔值，显然不能存在这样的 V_{name} 和 V_{-name}，他们的值都为真。同时我们定义符号“→”，$V_1 \rightarrow V_2$ 表示如果 V_1 为真，则 V_2 为真，并且称 V_2 为 V_1 的后继。

于是我们很自然得到，(N_1 => N_2)成立的条件是($V_{N_1} \rightarrow V_{N_2}$)，($N_1 | N_2$)成立的条件是($V_{-N_1} \rightarrow V_{N_2}$)且($V_{-N_2} \rightarrow V_{N_1}$)，这样，我们处理的任务中的所有要求(不包括几个特例)，都可以转化($V_1 \rightarrow V_2$)这种形式，也就是在图中连一条从 V_1 到 V_2 的有向边。下图所示的为(A=>-B)及(C| D)的情况：

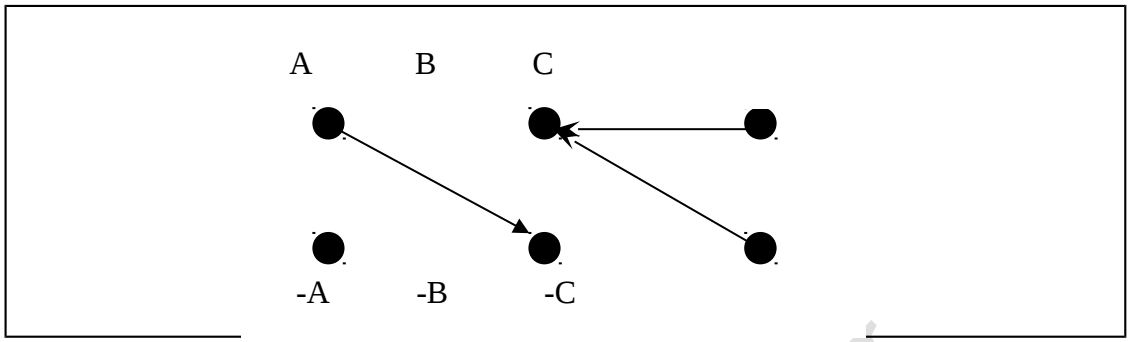


从定义 $V_1 \rightarrow V_2$ 可以看到，一旦 V_1 为真，那么 V_1 的所有直接后继和间接后继都必须为真。这样，我们的目标就是找到一个顶点集合 V，满足：

1. 如果 $V_0 \in V$，则 V_0 的所有后继组成的集合 $V_{son} \subset V$；
2. 不存在这样的 V_{name} 和 V_{-name} 同属于 V。

对于每一个 *name*，我们可以先试探 V_{name} 为真，如果其间接后继中没有 V_{-name}，则说明 V_{name} 为真是可以满足题意的；否则我们再试探 V_{-name} 为真，如果其间接后继中没有 V_{name}，则说明 V_{-name} 为真是可以满足题意的；如果试探均失败则无解。事实上，题目已经明确指出至少是存在一个解的。

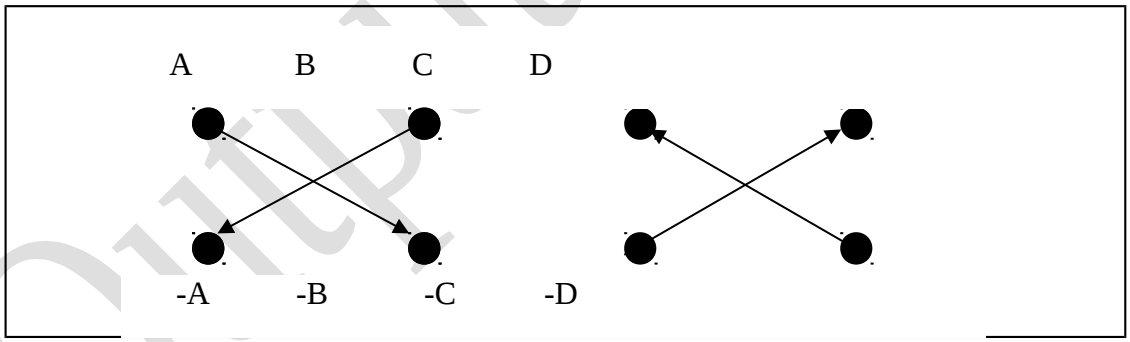
但我们可以清楚地看到，上述算法是有后效性的。我们完全可能在选中(注意这里不是试探性地选中，而是完成了试探的真正选中) V_i 之后，当前又必须经由某个 V_j 选中 V_{-i}。下图即是一个反例：



我们开始"顺利"地选中 A 与-B 以后，却在选 C 的时候面临无论如何都要选 B 的局面。这样，就造成了既选 B，又选-B 的错误局面。事实上，我们令 $S=\{-A,B,C\}$ 或 $S=\{-A,B,-C\}$ 都是满足题意的。这说明开始时我们很自然的构图方法是不正确的。

为了解决后效性问题，我们重新构图。

$(N_1|N_2)$成立的条件依然是$(V_{-N_1}\rightarrow V_{N_2})$且$(V_{-N_2}\rightarrow V_{N_1})$，而$(N_1 => N_2)$成立的条件我们从$(V_{N_1}\rightarrow V_{N_2})$扩充为$(V_{N_1}\rightarrow V_{N_2})$且$(V_{-N_2}\rightarrow V_{-N_1})$。于是前面的(A=>-B)及(C| D)的情况变为：



后面的内容与前文一致。下面我们就来证明这种构图是没有后效性的：

根据前文所述，$(N_1 => N_2)$成立的条件是$(V_{N_1}\rightarrow V_{N_2})$且$(V_{-N_2}\rightarrow V_{-N_1})$，

$(N_1 | N_2)$成立的条件是$(V_{-N_1} \to V_{N_2})$且$(V_{-N_2} \to V_{N_1})$。两者在本质上是相同的，即$(-N_1 => N_2)$等同于$(N_1 | N_2)$。也就是说，我们可以将它们都视为$(N_1 | N_2)$。

反设我们在选中(真正选中)V_i之后，当前试探中必须经由V_j选中V_{-i}，这就表明存在$(V_j \to V_{-i})$这条弧。根据构造的对偶性，必然还存在$(V_i \to V_{-j})$这条弧，由于V_i是已经肯定选中了的，那么其后继V_{-j}也必然选中，于是当前既选中了V_j，又选中了V_{-j}，这是不可能的，矛盾。所以问题不存在后效性

于是 Party 就算解决了，我们再回过头来看 Peaceful Commission 和 Excursion。

Peaceful Commission：

点 *i* 与点 *j* 不相容也就意味着必须满足$(V_i \to V_{-j})$且$(V_j \to V_{-i})$，反推得到$(V_{-i} | V_{-j})$，就是本题的$(N_{-i} | N_{-j})$。于是 Peaceful Commission 与本题实质上完全一样。

Excursion：

前文已经说明其模型是要满足“对任意一个前面给出的整数对*(i、j)*，点 *i* 和点 *j* 中至少有一个属于集合 *S*”，这就已经是我们的$(V_i | V_j)$，也就是本题的$(N_i | N_j)$。于是 Excursion 与本题实质上完全一样。

下面我们来分析算法的复杂度。

空间复杂度：线性，可以忽略。

时间复杂度：枚举 *N* 个点，每次枚举最坏的情况遍历但不重复遍历 *M* 条弧，所以时间复杂度为 *O(M×N)*。但如果确定了点 *i* 属于集合 *S*，那么V_i的所有后继就都必然属于集合 *S*，所以实际复杂度远小于 *O(M×N)*。

需要指出的是，这三个问题的 2-SAT 模型是存在 ***O(M)***算法的，不过比较复杂，限于篇幅，此处从略。

Party 小结：

本题的核心内容 2-SAT 出得比较好，但本题的价值远不止此。它启示我们，当今的题目已不能单单从算法的角度来衡量了，对于本题，这样得到的结果无疑是 NPC 问题。但事实上这道题我们得到了解，而且是在相当短的时间内。这道题

最完美的算法其实上是最笨的算法，最聪明的算法就是要求我们具体问题具体分析，在最短的时间内得到正确的结果。

本题目的精妙之处主要在于 10 个数据各具特点，“具体问题具体分析”，不要再以常规题目所有数据统一对付的思想来解决结果提交类问题，而是应该采用灵活多变的手法。对于一道题，我们可以使用各种各样的手段，但目标只有一个：快速得到正确结果！

此外，我们分析得到 Party 与 Peaceful Commission，Excursion 在本质上是完全一样的，在不到两年的时间里，同样的 2-SAT 模型屡次并且甚至变为结果提交的形式出现，这也需要引起我们的重视。

三、随机化

值得说明的是，这里的随机化和经典问题中的随机化有所不同。经典问题中的随机化需要证在规定时间内出解，且评测时只会运行一次。但结果提交类问题的随机化没有任何限制，目标只有一个：得到正确结果！

[例三]Tetris(NOI 2002)

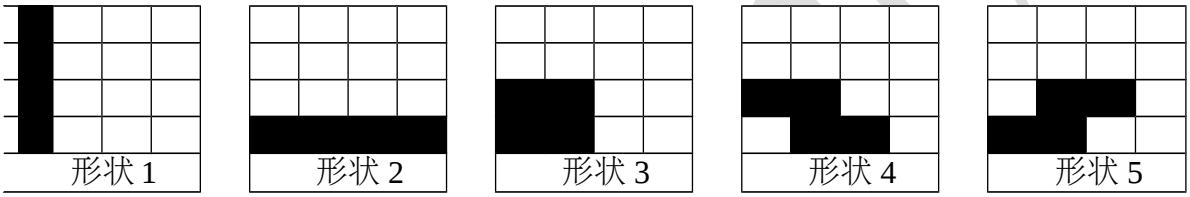
题目大意:

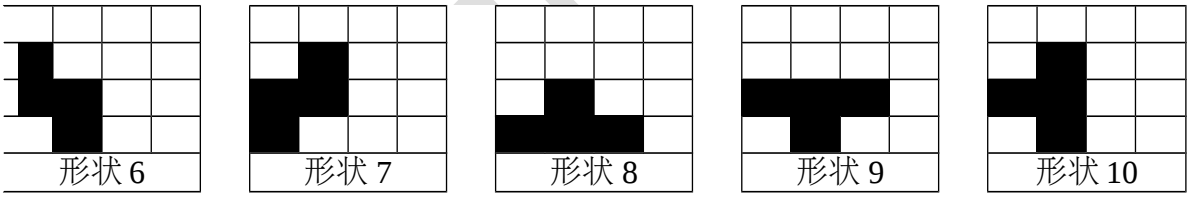
本题是关于俄罗斯方块的问题，只是本题所涉及的所有状态中，不能有悬空。

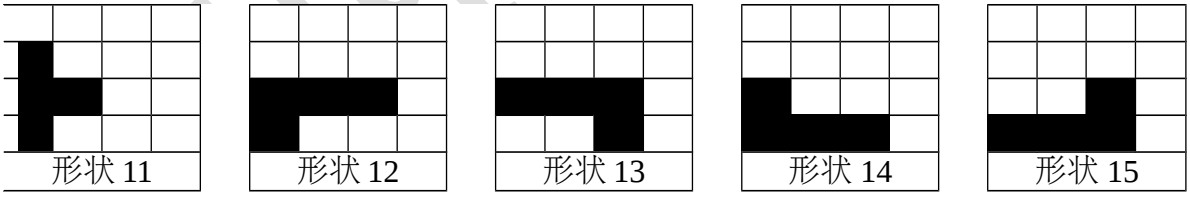
给出一个初始的棋盘状态，我们每次可以从标准的 19 种形状中任选一种，放到任意的位置上，只要不悬空，不超出边界即可。

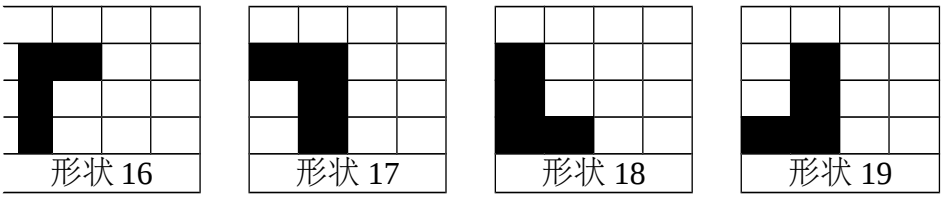
你的任务是在 100000 步以内将棋盘消空。输入数据保证有解。

题目的测试数据中最大的棋盘状态有 1202 列。









分析：

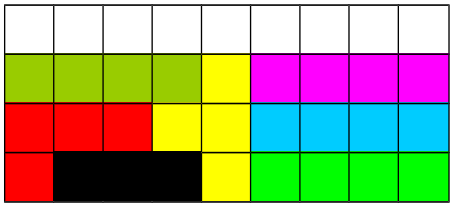
我们依然是先看测试数据。

Tetris1.in

9

0 1 1 1 0 0 0 0 0

如下即可解决，黑色为初始的格子。

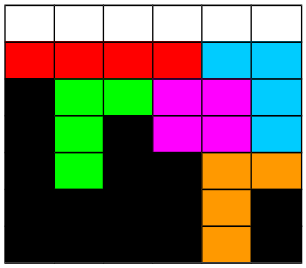


Tetris2.in

6

5 2 4 3 0 2

依然非常简单，我们一眼即可看出：



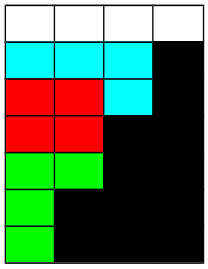
Tetris3.in

200

0 2 4 6 6 8 10 12 12 14 16 18 18 20 22 24 24 26 28 30 30 32 34 36 后略。

我们可以看到，数据以四列为单位有规律的出现。(0，2，4，6)，(6，8，10，12)，(12，14，16，18)……

对于每一个循环单位(a_i，b_i，c_i，d_i)(i=2，3，……50)，都是($a_{i-1}+6$，$b_{i-1}+6$，$c_{i-1}+6$，$d_{i-1}+6$)，而最基本的(0，2，4，6)是非常容易解决的：



所以本数据只需要用一个小程序即可解决。

Tetris4.in
16
8 1 7 0 5 6 1 0 1 5 2 0 3 7 6 0

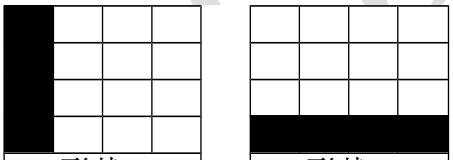
Tetris5.in~Tetris10.in
略

综合分析：
数据一、二：规模很小的数据，手工能在短时间内出解。
数据三：　规模很大，不过规律很明显，编一个很短的程序即可出解。
数据四、五：规模不是很小，也可以手工算，但需要一定时间，如手算的方法不好则很有可能算不出结果。
数据六…十：规模较大，规律很不明显或没有规律，手工已经完全无法胜任。

到这里，便有一个策略问题，人们总是习惯于在看到数据一、二、三等到手的分之后便匆忙动手，虽然得了 30 分，但浪费了很多时间。如果最后编出了理想的程序，那这里贪分就很不划算了。这也是命题者的意图所在，无形中考察了选手的比赛策略。

值得一提的是，对于数据 1，2，4，5，我们甚至可以编一个模拟程序，这样可以极大提高手算速度。一方面，因为辅以简单程序，数据分析效率会大大提高；另一方面，依然是我们处理结果提交类问题的核心：不择手段，只求结果。

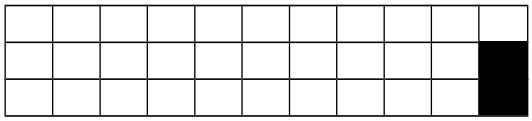
下面先谈一种可行的算法。玩儿过俄罗斯方块的人都知道，下面两种形状是最受欢迎的：



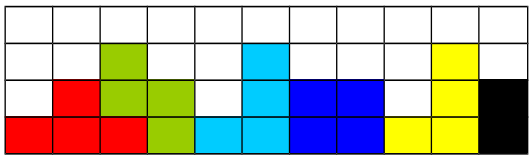
我们很容易发现，对于任何一个初始状态，可以仅用这两种形状，使棋盘成为高度不超过 3 的很矮的棋盘。接下来，我们只需用随机化算法将棋盘弄平。随机化的核心思想和实现都很简单，就是从左到右找“空”，然后随机用一个合法的形状填上此空，直到将棋盘完全弄平。

当然这个算法是很粗劣的，数据大一点就无能为力了，为此，我们还可以进行许多优化，主要是在随机的基础上加入一些人的思想。由于本问题的数据是给出了的，程序的优化就显得相对更加容易。

优化一：上述算法在棋盘“相当平”的时候，却在做大量在人看来非常愚蠢的操作。对于下面一个残局，



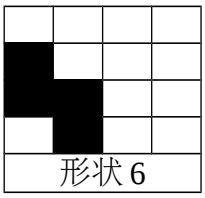
我们的算法却在这样随机生成形状：

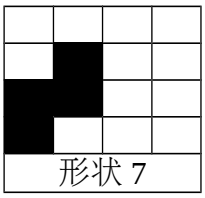


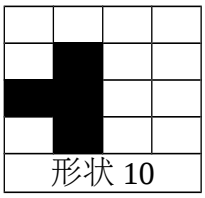
这样虽然合法，但是难以对付较大的数据，因为步数太多了。其实只需要：

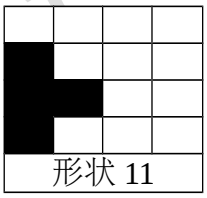
所以在可能的情况下，我们应该尽量使用形状 2 和形状 3，这样会使棋盘尽量显得平整。

优化二：同时，我们也应该尽量避免下面 6，7，10，11 这四种形状，这样就不会过多引入那些突兀。


形状 6


形状 7


形状 10


形状 11

至此，虽然算法还比较粗糙，但对付本题已绰绰有余了。类似的人工因素还有很多，比如还可优先考虑形状 9 等等，这里就不再继续讨论了。此算法的优势在于其思考及实现的花时非常少，程序简短，劣势在于出解步数较大，最大的数据需要一万多步才能出解。虽然对付本题的数据没有问题，但数据再大就显得有些无能为力。算法的时间复杂度为 *O(M)*，*M* 为游戏的总步数。

顺便值得一说的是，该随机化算法在一般情况下是可以瞬间出解的，但偶尔会由于程序自身缺陷而陷入随机陷阱，造成“死机”，而我们只需要终止程序即可。这其实也是结果提交类问题带来的好处，那就是只管结果，不管过程，无论中途发生什么，只要输出正确结果。同时，这种多次运行的随机化思想还可以更有助于我们逼近最优解，这在后面将要提到的 Xor 中是非常有意义的。

除了随机化算法，我们还有两种较好的算法：

算法一：我们每一步都尽量使棋盘变得更平，直到最后达到完全平整。具体来说，对于任意个棋盘 C，我们定义 $F(C)=\sum_{i=1}^{n-1}\left|H_i-H_{i+1}\right|$，其中 H_i 表示棋盘第 i 列的高。我们每一步的决策就是从当前棋盘 C 的子棋盘集合 S 中找出一个

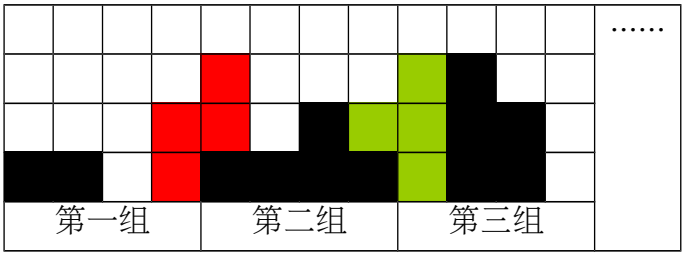
$C' \in S$，使得不存在$C'' \in S$，满足$F(C'') < F(C')$。如果有多个C'存在，我们可以随机[1]选取一个。容易看出，本算法的时间复杂度为$O(M \times N)$，其中M为游戏总步数。这个算法是很容易实现的，不过时间复杂度高了些。

算法二：构造。

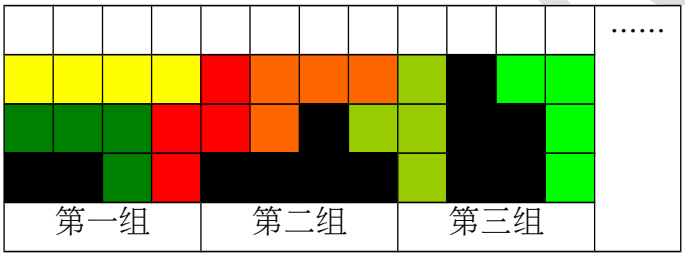
粗略描述如下：我们从左到右以4列为单位对棋盘进行分组，最后剩下的如果不足4列，则单独为一组。我们设法先将每组填平，然后再用形状2将整个棋盘填平。

但有些组我们是无论如何都无法填平的，那就需要我们在初识化时在相邻两组交界的地方放上某些特定的形状，使得该组的方块数为4的倍数。由于输入数据是保证有解的，所以这里的"该组"不包括未满4列的最后一组。

如下图所示，对于第一、二、三组，通过组内调节是无论如何都无法填平它们的，因为他们原有的块数分别为2、5、5，都不是4的倍数。我们可以通过下面的组间调节使他们都变成4的倍数，调节之后三个组的块数分别变成4、8、8。



之后每组的填平方法可以使用算法一。如下图所示：



此算法的时间复杂度大约为$O(M)$，不过实现起来要麻烦些。

下面我们来将这三种算法做一下比较。由于三种算法的空间复杂度都是$O(N)$，在比较的时候，忽略此项指标。

从上表我们可以看出，对于结果提交类问题，随机化算法和算法一是最合适的，如果非结果提交，则算法二最好。不同的要求下，我们采取的算法可能是不同的，因为还要综合考虑算法实现的复杂度。

	时间复杂度	运行速度	算法实现的复杂度	解的质量
随机化算法	$O(M)$	<1s	低	较差
算法一	$O(M \times N)$	<10s	低	较好
算法二	$O(M)$	<0.1s	高	较好

[例四]Xor(IOI 2002)

题目大意：

有黑白二染色的 $N\times N$ 格子，我们每次操作先选择一个矩形，然后将此矩形包含的所有格子颜色取反。现要求用尽可能少的操作次数使这些格子全部变为白色

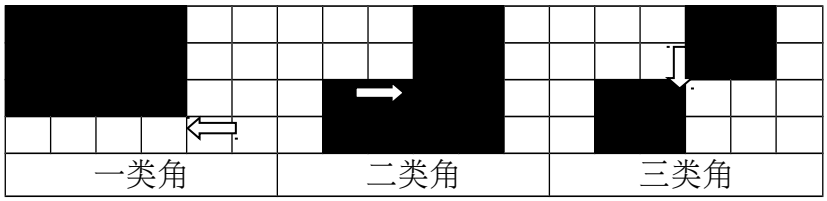
给出十个输入文件，其中 $N\leq 2000$。

每个数据得分的计算方法：

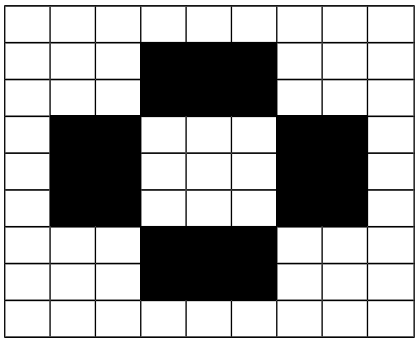
$$1+9\times\frac{\text{该数据所有选手中最少的操作次数}}{\text{该数据你的操作次数}}$$ 。

分析：

这类题目是比较普遍的，经验告诉我们应该首先考察长方形的“角”。角可以分为三大类：



显然，一、二类角必是某次操作中的长方形的角，而三类角则未必是。如：



此位型只需要两次操作即可完成，其中的四个三类角都不是某次操作中的长方形的角。

于是我们的算法也就基本成型了：

1. 将所有的一、二类角做成集合 A。
2. 从 A 中找出所有能组成矩形四个顶点的角组(a_i，b_i，c_i，d_i)，分别视为矩形操作并一一删去。
3. 从 A 中找出一个能组成某个矩形三个顶点的角组(e_i，f_i，g_i)，设缺的顶点为 h_i，将(e_i，f_i，g_i，h_i)视为一次矩形操作，在 A 中删去 e_i，f_i，g_i，并添上 h_i。
4. 如果 A 为空，则中止，否则转 2。

算法的正确性容易证明，此处从略。

该算法其实上是贪心的，由于题目只要求我们求较优解，所以算法是可行的。每次四点组的删除会使|*A*|减少 4，每次三点组的删除会使|*A*|减少 2，所以|*A*|在算法中总是递减。最优解在最理想的情况下每次操作都能删去一个四点组，而最坏的情况我们也能每次使|*A*|减少 2，因此该算法在最坏的情况下每个数据至少可以得到 1+9*2/4=5.5 的分。当然在实践中，得分远不止此。

在算法的实现上，我们可以引入一个 *(N+1)×(N+1)* 的布尔数组 *Couple*，*Couple[i,j]*为真当且仅当在当前的图形中存在*(a,i)*，*(a,j)*两个角，而它们都属于一二类角集合。

引入随机算法优化:

我们可以看到，2 的处理是无所谓先后顺序的，而 3 的选择则直接关系到最终解的优劣。然而问题的规模 2000*2000 非常大，即使是很简单的选择策略在时间上都难以承受。因此我们完全可以随机选择我们需要的三点组，每次选择三点组+删除+转回 2 反查四点组的总时间复杂度为 *O(N)*，这样程序主体的时间复杂度就降至完全可以忍受的水平了，如果最后得到的操作数为 *M*，则程序主体的时间复杂度就为 *O(M×N)*，从实践中我们看出，*M* 最大也不超过 1000，时间主要花在预处理上面。除了速度，我们在后面还可以看到，解的质量也是可以接受的。

然而我们并不能保证次算法每次都能够得到相当优秀的解，虽然相差很小。但是在本题的评分方法下，这种差异对最后得分的影响还是不容低估的，于是我们选择多次运行取最佳解。我想这也应该是命题者将此题作为结果提交类问题的原因之一，考察程序的最佳表现，而不是随机表现。

下面是该算法的表现和目前已知的最优解(当然不包括此程序)比较的情况:

	Xor.pas 的解	已知最优解	得分
Xor1.in	3	3	10.0
Xor2.in	35	33	9.5
Xor3.in	28	27	9.7
Xor4.in	110	113	10.0
Xor5.in	74	74	10.0
Xor6.in	487	469	9.7
Xor7.in	515	507	9.9
Xor8.in	200	200	10.0
Xor9.in	500	500	10.0
Xor10.in	845	841	10.0
总分			99

注：以上结果是 10 次运行得到的。

结论:

本题用随机化的方法是相当有效的，特别是针对其结果提交的特点实行多

次运行，使得有 5 个数据达到甚至超过了已知最优解，其它数据的结果也都非常接近已知最优解。当然，这种算法的实现要比其他的要容易得多——也就是，我们在尽可能短的时间内得到了较为满意的结果。

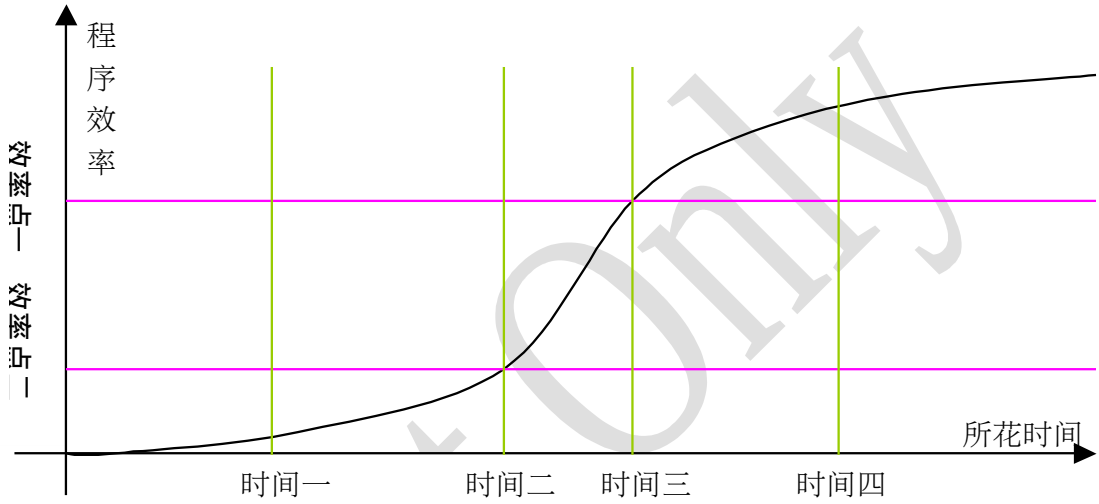
当然本题还有很多算法，比如匹配等，都是比较稳定而且质量较高的算法。这也再次说明结果提交类问题由于其开放型，一般都有多种方法，同时也需要我们能够优中选优。

四、结果提交类问题的策略

正如前文所提到，结果提交类问题对选手的考察已不再局限于算法和数据结构，而是涉及一些更广泛、更深层次的东西，如比赛策略等。下面我们就来讨论一下该类问题所带来的不同的比赛策略。

1、 不同的效率观和策略观

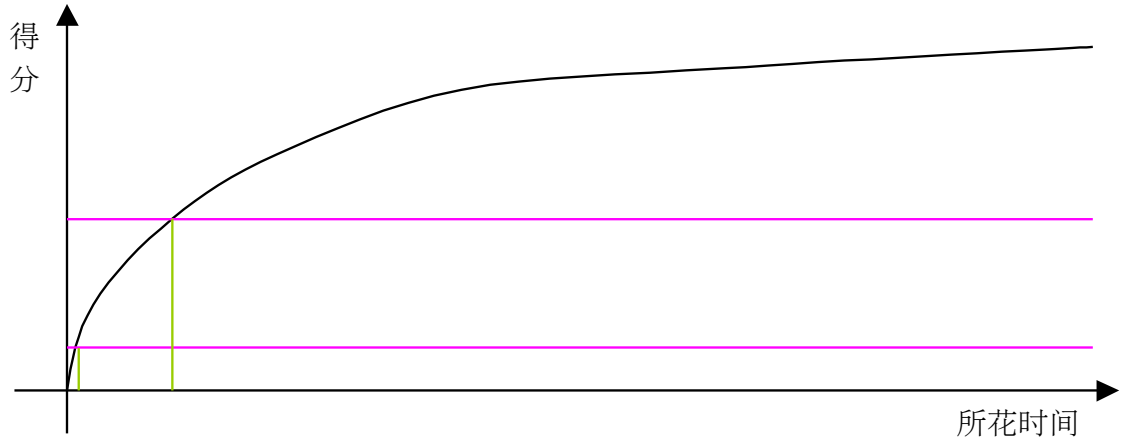
前文已经提到，结果提交类问题效率观的改变首先体现在对算法的衡量标准不再仅是时空复杂度，还要考虑算法的实现，考虑如何在最短的时间内得到正确结果。让我们先来看看经典的完成题目所花时间与程序效率关系图：



这种模式用“厚积薄发”四个字来形容最恰当不过了。而且可以发现：

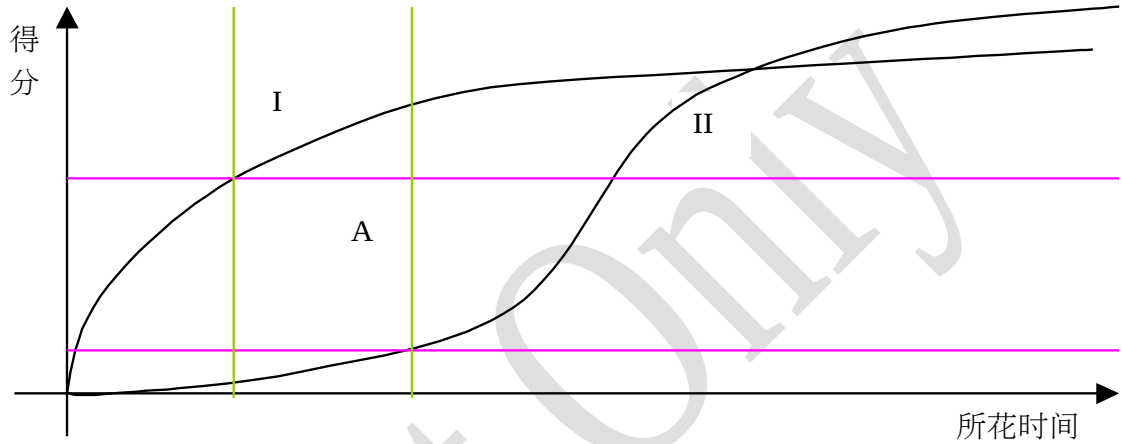
1. 我们不知道什么时候该“适可而止”。我们时常在时间四停止，但事实上，可能在时间三或更早时已经达到题目要求，从算法角度讲，我们得到了进一步的提高，但从策略角度讲，是不明智的。
2. 时间二道时间三这较短的时间内，我们的程序效率有了很大的提升这对于最后的结果，往往是起着决定性的。注意这里的时间一和时间二离原点都比较远，在开始的相当长一段时间里，我们程序效率的提升很慢，主要是由于在设计算法和数据结构，根本还不涉及程序的实现。

然后我们再来看看结果提交类问题的所花时间与得分关系图，由于结果提交类问题不同的数据处理方法不一定相同，我们以难以用程序效率来衡量，所以后面我们统一使用得分来说明。



这种关系图是比较符合结果提交类问题的特点的。开始时一般都可贪分，所以图像斜率较大，后面斜率逐渐减小，以至最后得分的进展很不明显。当然，既然我们有测试数据，我们可以在心中对能够得的分数有一个比较准确地估计，所以决不会像经典问题那样多花无谓的时间。然而和前面经典问题不同的是，较短时间内使我们的程序效率有了很大提升的时间段(两条绿线之间)前移了不少。这就使得我们如果中途放弃，代价会相对较低。

下面我们来综合分析一下两种题目：



图中区域 ***A*** 是我们应当把握的关键，突出表现为曲线一速度快，效率(得分)高。对于同样的得分(水平红线)，我们可以看到，两者在完成时间上有着可观的差异。正如前文所说，结果提交类问题完成质量的标准是完成时间！所以曲线 I 的优势是无容置疑的。此外，在比赛中，由于心态、策略或其他原因导致我们中途放弃某道题的时候(竖直绿线)，曲线 I 的优势则更为明显。

结果提交类问题对选手策略的考察，可以体现在题目的时间策略上及该试试题的全局策略上。我们可以花较短的时间就对能得的分数有相当准确地估计（如 Birthday party），这是前所未有的。因此，如何在理想的时间内得到得分/时间的最佳值，如何舍弃部分数据以换取时间完成其他题目，如何有效快速的贪分，如何解决数据间方法的重叠，等等，都需要我们思考。

由此可见，结果提交类问题所带来的效率观是崭新的，倘若我们还停留在仅仅以算法的效率来衡量程序的层面上，那我们势必吃大亏。

2、 不同的时空复杂观

可以说，结果提交类问题完全打破了经典的时空复杂观和程序优化模式：

1. 大大扩展了程序可用的空间，没有严格限制；
2. 时限变得更加自由，甚至可以多达几个小时；
3. 由于近年来 Linux 在竞赛中的引入，多线程的方法也开始崭露头角，我们甚至可以因此将题目的时限扩至 5 个小时。

以前，我们只有少量的空间+很短的时间+单一的线程，而现在，我们有了测试数据+充足的空间+相当长的时间+多个线程。我们已没有理由再用原来的衡量标准和原来的思路来看待现在的问题，我们需要的只有一条：得到正确的结果，而不用管过程。当然问题应该一分为二地来看待，追求结果并不等于放弃对过程对完美算法地追求，而是应该将实力与巧力有机结合起来。

让我们再来看一个例子。

[例五]Double Crypt(IOI 2001)

题目大意：

任给一个信息块*p* (纯文本plaintext)和一个密钥字块*k*，通过AES的加密函数*E*，便会得出一个加密后的块*c* (加密文本-ciphertext)：*c=E(p，k)*。

AES加密函数*E*的反函数是解密函数*D*，它满足

D(E(p，k)，k)=p，　　*E(D(c，k)，k)=c* 。

在Double*AES*双重加密法中，先后使用两个独立的密钥字块 k_1 和 k_2，先使用 k_1，然后用 k_2：　$c_2 = E(E(p，k_1)，k_2)$。

题目给出另一个整数*s*，使得 k_1 和 k_2 都属于集合{ $1,2,3.....2^{4\times s}$ }

提供给你*p*和 c_2，求出 k_1 和 k_2。

输入文件有三行，分别是*s*，*p*和 c_2。

输入文件：

数据	*s*=	*p*=	c_2=
一	1	00112233445566778899AABBCCDDEEFF	6323B4A5BC16C479ED6D94F5B58FF0C2
二	1	3FCFAA530B83FAC26C3AF18C958F0665	68F4005C13000809D9F5F090C36FD0F2
三	2	200120012001200120012001200120012001	9B4AF5F399003A4AFF6D1FC0743E34FF
四	2	CBF1F1840B32AD1FF76A86FAE9034526	200120012001200120012001200120012001
五	4	7F5471276E955708D0BF998EB40E99AA	9E68E681898349BC4ECAB33DC530E9B9
六	4	7F5471276E955708D0BF998EB40E99AA	FB8E04CEE94601A32F30E35FC18C54C9
七	4	7F5471276E955708D0BF998EB40E99AA	CB4AFEC5C0F4A1C381F014E740A2E508
八	5	535A0B970CAF68E68F541AB12636FA7F	200120012001200120012001200120012001
九	5	200120012001200120012001200120012001	9B4AF5F399003A4AFF6D1FC0743E34FF
十	5	FA101AB101BC101CD101DE101EF101FA	3466A8C34427239F2AD9E0DE3188A8E3

简要分析：

很显然，本题用正反两个方向查找，然后用Hash表来比较的方法是比较理想的。如果不考虑冲突，则时间复杂度为$O(2^{4\times s})$，但实现是需要花一定时间的。这种算法比较简单，此处就不赘述。

如果我们退一步，采用时间复杂度为$O((2^{4\times s})^2)$的枚举的方法，并注意到数据三和数据九输出应该完全一样(p和c_2完全相同，只是s不同，但不影响结果)，数据五六七可以一起枚举等这些命题者有意用来考察选手观察能力的数据特点，我们完全可以在非常短的编程时间内得到80%以上的分数。当然，程序运行时间很长，不过我们可以通过另一个线程来完成。虽然并不完美，但符合该类问题的特点，充分利用了时间和线程优势，为其他题目节省了大量的时间，这也是符合结果提交类问题的策略的。一言蔽之，就1h+1h+1s<1s+0.5h+0.5h[2]！

3、 手工运算与程序运算的协调策略

许多结果提交类问题的精妙之处，就是使手工运算与程序运算巧妙地结合了起来，如 Crack the code，Birthday Party 等都是这样。这样不仅将我们的脑力更加深层地更加高效地植入了程序之中，而且使任务的实现变得简洁迅速。

很显然，我们既不能在 Tetris 中大量依赖手算，也不能在 Party 中完全依靠程序运行，倘若不能很好地将二者有机地结合起来，往往不仅浪费了时间，而且还有可能导致无法出解。

正所谓智者善用物也，我们只有充分利用所有能够利用的主客观条件——大脑与计算机，才能在最短的时间内得到正确的结果。

【总结】

结果提交类问题的出现，使我们不能再以传统的观点和方法来处理此类问题，而是应该具体问题具体分析，从全局的角度设计最好的方法来得到需要提交的文件。我们只有善于观察、灵活分析、巧妙设计，同时，将手工与程序完美结合起来，才能快速准确地解决任务。

一道结果提交类问题，我们既可采用经典方法，又可以采取新的方式来完成，正如 Crack the code 等，同时也可能经典方法完全无法对付，如 Birthday party 等。同样的题目，是否以结果提交的形式出现，可能决定我们采用什么算法更好，同时各种方法各有优劣，各有适用范围。于是常出现以往我们觉得很漂亮很精妙的算法在结果提交类问题中就显得相形见拙了，因为我们的衡量标准已经改变。

总而言之，我们可以用一句话概括本文，那就是：永远提醒自己，你所需要的只是在最短的时间内得到正确结果，而不是过程。

奥林匹克的精神是什么？更高、更快、更强。
结果提交，让我们站得更高，做得更快，变得更强。

【附录】

[1]这里虽然也是随机，但这里的随机只是一个小模块，显得远没有前面的随机那样纯粹。
[2]在大多数情况下，这时间显得太多了，绝大部分程序都还是非常快的。

【参考文献】

1. The 7th Baltic Olympiad in Informatics BOI 2001 Tasks and Solutions
   By Marcin Kubica
2. Summary of the 9th Central European Olympiad in Informatics
3. 第十九届全国信息学奥林匹克竞赛 NOI 2002 试题
4. IOI 2002 试题
5. IOI’01 Competition (Second Edition)
   By Jyrki Nummenmaa， Erkki Makinen and Isto Aho