第46届ICPC世界总决赛解决方案草图

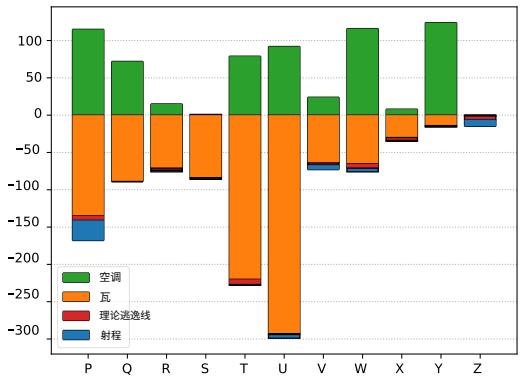
免责声明这是对第46届ICPC世界总决赛可能解决问题的一些方法的非官方分析。这些分析并非旨在提供完整的解决方案，而是为了概述一种可以用来解决问题的方法。如果您对下面提到的一些术语或算法不熟悉，您最喜欢的搜索引擎应该能够提供帮助。如果您发现任何错误，请发送电子邮件至per. austrin@gmail. com告知。

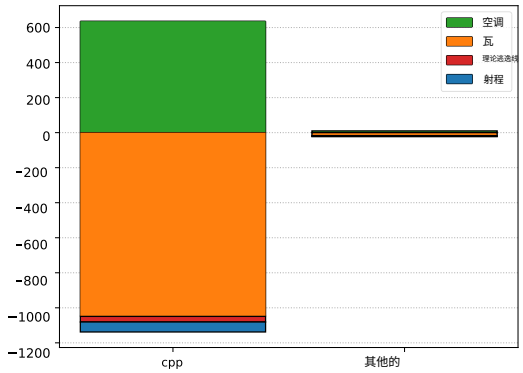
-致Austrin、Arnav Sastry、Paul Wild和Jakub Onufry Wojtasz-czyk

摘要

祝贺北京大学
作为第46届ICPC世界冠军的称号！

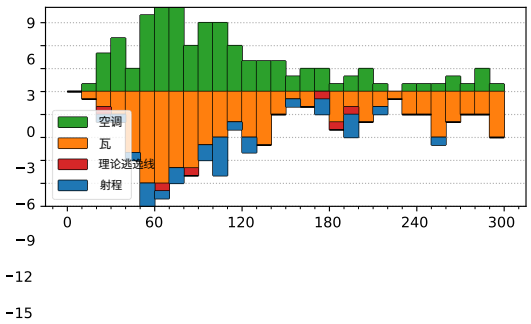
以下是两个图表，分别显示了每个问题和每种编程语言的提交数量。正y轴表示接受的解决方案数量，负y轴表示被拒绝的解决方案数量。





C++是迄今为止最主流的语言。在第46届和第47届世界总决赛（同时举行）中，大约1%的提交是用Python编写的。5%的提交使用了Kotlin，其余98.5%的提交使用了C++。没有Java的提交。关于解决方案大小的说明：以下是每个问题最小裁判和团队解决方案的大小。这些数字仅用于提供数量级的指示。裁判的解决方案并非为了最小化而编写（尽管我们中的一些人可能会过度压缩代码），通过删除空格、重命名变量等方法可以轻松缩短代码。当然，参赛队伍在比赛期间编写的代码也是如此！活动图解释：下面展示了每个问题的团队活动图。该图显示了比赛过程中不同类型提交的数量：x轴表示时间（分钟），正y轴表示接受的解决方案数量，负y轴表示被拒绝的解决方案数量。

问题P：变红问题作者：Jakub Onufry Wojtaszczyk和Arnav Sastry被115支队伍解决。
首次解决时间为19分钟。
最短的团队解决方案：1691字节。
最短的法官解决方案：1341字节。



这是一个相对简单的问题。让我们将三种颜色建模为红色=0，绿色=1，蓝色=2。然后按下按钮的效果是所有由该按钮控制的灯都会增加1，模3。

设xi为按下按钮i的次数（未知）。那么，由按钮i和j控制的灯?必须变红的要求变为方程c?+xi+xj=0（模3），其中c?表示该灯的初始颜色。请注意，如果已知xi或xj中的一个，这个方程可以立即计算出另一个的值。这意味着，如果我们考虑一个隐含图，其中两个按钮i和j如果控制相同的灯则相连，那么根据一个按钮的值xi，我们可以传播并计算出该按钮整个连通分量内的所有值。

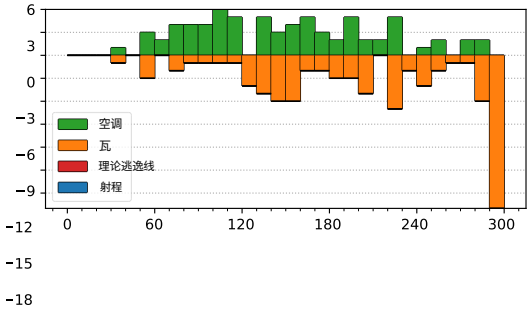
这导致了以下线性时间算法：对于图的每个连通分量，选择一个任意按钮i，尝试所有3个可能的值xi = 0,1,2，并传播结果。如果发现不一致（某个方程未满足），则该xi值无效。否则，检查总按键次数（分量中xi的总和）。如果所有3个可能的xi选择都无效，则没有解；否则，选择导致该分量内最少按键次数的那个。

在问题中，也可能有一些灯由单个按钮控制，导致方程形式为c?+xi=0（模3），这立即确定了xi。可以将这些情况特殊处理并先传播其值，但更稳妥的做法可能是直接将它们包含在上述算法的不一致性检查中。

问题Q：执行容器洗牌

问题作者：
Matthias Ruhl和Derek Kisman

由72个团队解决。
32分钟后首次解决。
最短的团队解决方案：513字节。
最短的法官解决方案：407字节。



定义两个堆栈中仍存在的容器的“联合顺序”如下。假设第一个堆栈中的容器从下到上为a 1，...，ak，第二个堆栈中的容器从下到上为b 1，...，bl。那么“联合顺序”是a 1，...，ak，bl，...，b 1。

取任意两个集装箱i和j。请注意，在卸载过程中的任何时刻，即在i或j被装载到卡车上之前，i和j之间的联合订单区间包含着在装载开始前位于i和j之间的集装箱，减去已经装载到卡车上的一些集装箱。

现在假设我们有两个容器ai和ai+1，我们想知道在卸载ai+1之前需要移动多少个容器。答案正是这些容器的集合。

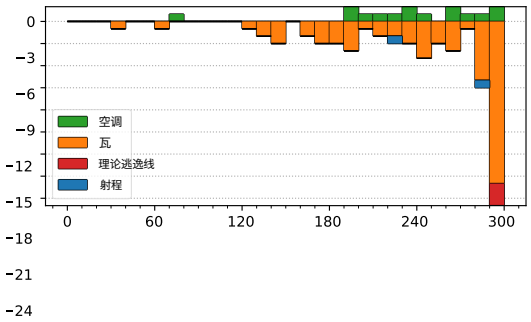
在联合阶数区间ai，ai+1。

因此，现在我们需要计算在尝试卸载ai+1时，处于联合订单区间ai，ai+1内的预期集装箱数量。我们希望检查某个集装箱v，在这个区间内的概率是多少。

如果v在ai之前卸载，概率为零。否则，概率等于v在装载开始时处于联合顺序区间内的概率。当v < min（ai，ai+1）时，该概率为0；当v > min（ai，ai+1）时，该概率为1/2。

实际上，对于单个容器，使用支持点更新和范围求和的数据结构，如范围树或Fenwick树，可以在O（log n）中计算出在ai之后装载的、小于min（ai，ai+1）的容器的数量。

问题R：动物园管理问题作者：Jakub Onufry Wojtaszczyk和Derek Kisman由15支队伍解决。首次解决时间为76分钟。
最短的团队解决方案：3475字节。
最短的法官解决方案：3163字节。



首先，我们可以观察到，所有桥都可以从图中移除而不改变结果，因为没有循环通过它们。这样就给我们留下几个独立的子问题，每个2边连通分量一个。

现在我们考虑一个每个边都是一个循环的一部分的组件。如果该组件是一个简单的循环，那么问题就变成了检查字符串的循环等价性，这可以通过任何线性时间的字符串匹配算法来完成。

如果一个组件包含多个循环，则答案可以是“任何排列都可以实现”或“任何偶数排列都可以实现”。这里我们省略了大部分证明细节，但证明此类命题的一般思路是首先找到一种方法，使任意两个标签进行一次交换，然后展示如何将此交换与给定的循环结合，以实现任意交换，从而实现任意排列。同样地，对于偶数排列，首先找到一种方法，使任意两个标签形成一个三元循环，再形成任意三个元素的循环，最后形成任意偶数排列。

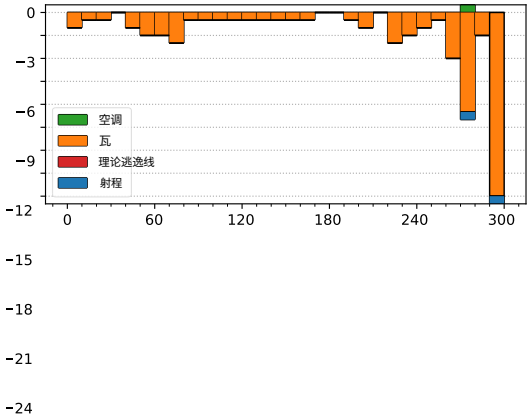
首先，考虑一个边属于多个循环的情况，或者等价地，考虑两个顶点由三条边不相交的路径连接的情况。通过沿这三个由选择其中任意两条路径形成的循环旋转，可以实现这个子图中标签的任何排列，也可以证明这允许实现整个组件中的任何排列。

如果一个组件由多个仅共享单个顶点的循环组成，那么我们可以选取两个相邻的循环，比如a和b，并执行置换aba-1 b-1，这会产生三个标签的循环。根据上述的一般策略，因此我们可以生成所有偶数置换。如果至少有一个循环的长度是偶数（因此是一个奇数置换），那么我们也可以生成所有奇数置换。否则，只能实现偶数置换。

问题S：弥合差距问题作者：

Walter Guttmann和Paul Wild

由1个团队解决。
276分钟后首次解决。
最短的团队解决方案：1433字节。
最短的法官解决方案：857字节。



关于溶液的结构，可以进行以下观察：

? 每当一个小组向后移动时，这个小组就只有一人
? 每当一个小组向前移动时，该小组要么是：

-最慢的k个人和最快的c - k个人，他们稍后会返回，或者

-一些跑得最快的人，他们稍后会回来，或者

-所有的人都离开了，假设他们不超过c人。

除了最后一个向前交叉之外，我们还需要进行一次向后交叉。因此，我们可以选择在向前交叉快速步行者时支付向后交叉的费用，并记录我们累计了多少次已支付的向后交叉。

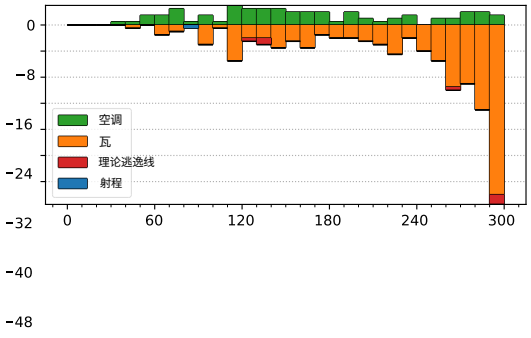
这自然转化为动态规划解决方案，其中我们计算出已经过境的k个最慢的人的成本，以及获得的l次回程。简单来说，这是O（n3）——我们有n2个状态，每个状态最多有n次转移。但是，如果你仔细观察，积累超过n /c次回交总是没有意义的（因为这已经足够让所有人交叉了），所以状态空间是O（n^2/c）。同时，从一个状态出发的转换次数是O(c)，因此在过滤掉不可达状态和不存在的转换后，实际成本是O（n^2）。

虽然最终的代码很短，但许多评委发现实现起来很棘手。

问题T：Carl的假期问题作者：

Arnav Sastry和世界总决赛评委由79支队伍解决。

33分钟后首次解决。
最短的团队解决方案：1805字节。
最短的法官解决方案：893字节。



显然，这个问题的挑战不在于运行时间，而在于如何以一种方便的方式表示三维几何问题。

在平面上，两点之间的最短路径始终是一条线段。因此，路径将是从一个金字塔的顶部到其底部的一段线，然后是穿过地面到达另一个金字塔底部的一段线，最后是从该金字塔的底部到顶部的一段线。

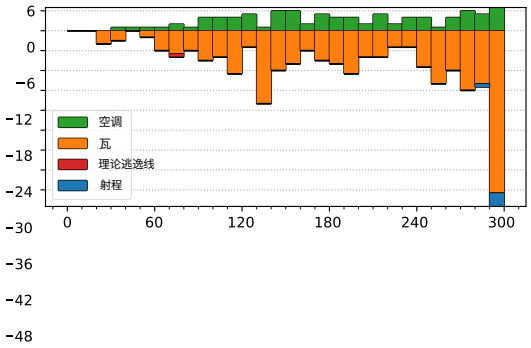
我们有16种方法来选择我们将要旅行的两个金字塔的面，所以我们将会检查所有这些方法。现在，我们有两个倾斜的三角形，每个都通过底边与地面相连，我们需要从一个三角形的顶点到另一个三角形的顶点，尽量走最短的路径。如果我们绕着每个三角形的底边旋转并将其压平，答案会是一样的——因此我们现在只需要找到平面上两点之间的最短路径。

条件是它通过两个指定的线段。平面上两点之间的最短路径就是一条线段。所以，总结一下：

? 我们遍历两个金字塔上的所有面的选择。
? 我们围绕包含选定面基部的直线旋转金字塔顶部，使金字塔顶部落在地面上。
? 我们检查两个旋转的顶部之间的线段是否以正确的顺序与两个底座相交。
? 如果是，我们把旋转的顶部之间的距离作为候选距离。
? 我们输出最小的候选距离。

另一种方法是观察到，对于一对金字塔面，我们可以用两个角度θ1和θ2来参数化蚂蚁所走的路径。可以通过黄金分割或三元搜索找到dist（θ 1，θ2）的最小值。

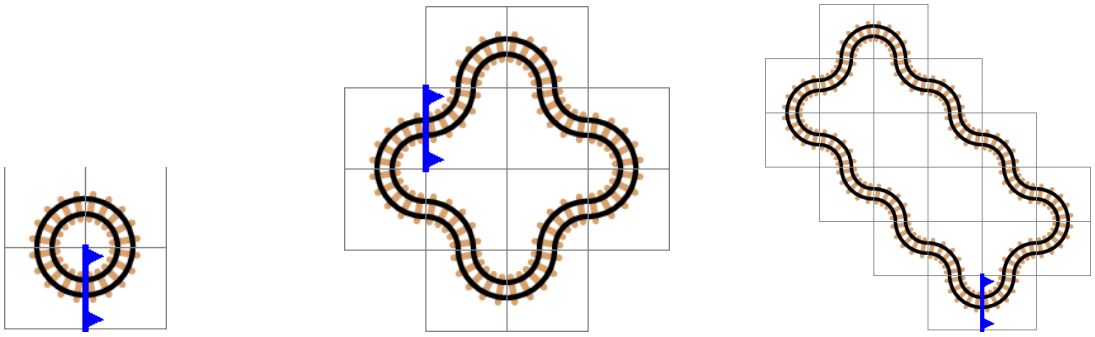
问题U：玩具火车轨道问题作者：Matthias Ruhl和世界总决赛评委由92支队伍解决首次解决时间为31分钟
最短的团队解决方案：668字节。
最短的法官解决方案：464字节。



像这样的建设性问题有许多不同的解决方法。下面是一位法官的解决方法。任何闭合回路都必须有偶数个弯曲轨道段和直线轨道段，因此我们可以先将两者都设为偶数。然后需要处理三种基本情况：

1、没有直线轨道段，2、c=0（模4），
或
3. $c \equiv 2 \pmod 4$.

如果没有直线段，我们可以制作图1所示的两种“ 基础 ”轨道之一。然后，你可以在图1b的两侧重复应用“ RL ”，每次扩展4个曲线，如图1c所示。请注意，无法制作恰好有8个弯曲段的轨道。



(a) 基线轨道：LLLL (b)基线轨道：LRRLRRLRRLRR (c) LRLLRLRLRLLRLLRLRLRL图1：全曲线情况。

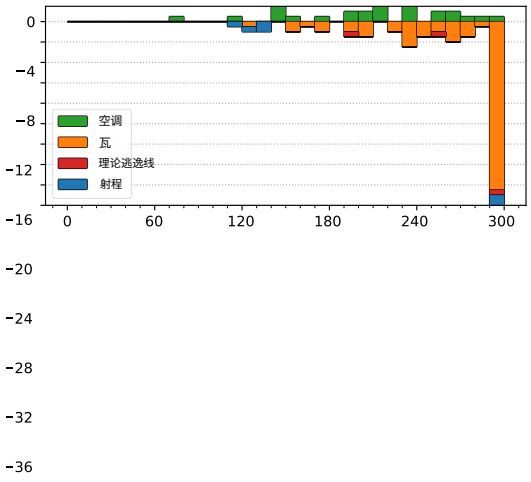
另一方面，如果有至少两条直线段，则我们根据c（模4）的余数进行分支。由于我们已强制c和s为偶数，因此这个余数要么是0，要么是2。图2显示了两种可能的“基础”轨道，我们可以在这两个图形中分别放置两条直线段，位于相对两侧，并且还可以加上上述的“RL”扩展。

(a) 左下锁骨　　(b) LSLSLLRL图2：直线轨道段情况下的两条基线轨道。

问题五：三种骰子问题作者：

Derek Kisman和Walter Guttmann

由24个团队解决。
76分钟后首次解决。
最短的团队解决方案：1545字节。
最短的法官解决方案：871字节。



考虑骰子D3的一面，其值为v。令S_i(v)，其中i∈{1,2}，表示当D3落在这一面上时，骰子Di获得的点数的期望值。骰子D1的整体期望值是D3所有面上S_i(v)的平均值。

S 1(v)简单来说就是D1中大于v的面的数量，加上等于v的面的一半。同样的规则也适用于S 2(v)。这意味着（S 1(v)，S 2(v)）对有O（n）种可能的值，我们可以在O（n log n）时间内轻松计算出所有这些值。现在我们正在考虑为这些点分配权重，使得S 1(v)的加权平均值≤0.5，而S 2(v)的加权平均值尽可能高（反之亦然）。

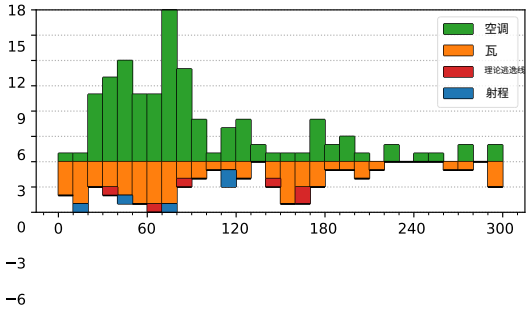
这是一个几何问题。请注意，通过一组点的加权平均值可以得到的所有点恰好是它们的凸包；因此，我们需要在凸包与x≤0.5的交集中找到最高的y值，这可以通过多种方法在O（n）时间内确定。

问题W：斯芬克斯之谜

问题作者：

Martin Kacer和Per Austrin

由116个团队解决。
8分钟后首次解决。
最短的团队解决方案：478字节。
最短的法官解决方案：347字节。



这是一个简单的交互式问题，解决方法有很多。对于这类问题（可能对当前情况有些过分）有一个普遍的观察。考虑一个5 × 3的矩阵

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \\ a_{51} & a_{52} & a_{53} \end{pmatrix}$$

第i行是您在第i个问题中给出的三个数字。然后，如果A的任意3行线性独立，我们就能唯一确定正确答案。这是因为如果我们移除一个真实答案，将会得到一个不一致的方程组；而如果我们移除谎言，就会得到一个过量确定的方程组。换句话说，我们可以唯一识别出哪个答案是谎言，然后利用其他任意三个答案来恢复正确答案。

既然我们可以自由地选择A，那么最好选择一个答案容易恢复的方式，例如。

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 2 & 3 \end{pmatrix}$$

请注意，我们不能将最后一个查询改为（1,1,2），因为这样最后三个答案就会线性相关，如果前两个问题中的一个撒谎了，我们就无法判断是哪一个撒了谎。同样地，（0,1,2）也不能作为最后一个查询，因为这样第二、第三和最后一个问题是线性相关的。

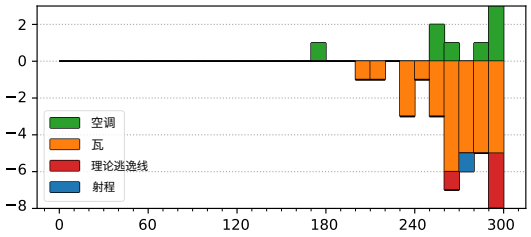
问题X：四重奏

问题作者：

Martin Kacer 和 Yujie An

由8支队伍解决。

179分钟后首次解决。最短团队解决方案：3096字节。
最短的法官解决方案：2894字节。



为了解决这个问题，我们需要理解单个动作能提供哪些关于世界状态的数据，并确定我们所掌握的信息是否与某些初始牌位相兼容。鉴于我们必须检查我们的知识的一致性，

对于可能的起始手牌，我们将用它对起始手牌的含义来表示我们的知识（并单独保留从我们开始时起该状态的变化）。

输入中列出了三种类型的事件，我们需要了解它们如何影响世界的状态。我们将存储以下类型的信息：

? 玩家P手中出现了一张特定的牌
? 玩家P的牌组中没有特定的牌
? 某个特定玩家在集合S中拥有一张尚未移动的牌

此外，对于每张卡，如果它移动了，我们存储它的当前位置。

如果玩家x向y要一张C卡，并且得到了它，那么：

? 如果我们不能确定x手中有某花色的牌（我们可能知道x现在手中有该花色的牌，或者他们手中有未知的该花色的牌），那么我们现在就标记x开始时手中有我们还不知道的该花色的牌。
? 如果我们知道C的当前位置，它要么在y手中，在这种情况下我们什么也学不到，要么在别人的手中，这意味着有人作弊了。
? 如果我们不知道C的当前位置，那么我们得知它最初在y手中，现在在x手中。此外，如果我们知道y手中有一张未知的花色牌，我们现在可以清除这些信息。

如果玩家x向玩家y要C，但没有得到，那么：

? 我们学到的和上面一样，x在花色中有一张牌，
? 如果我们不知道C的当前位置（这意味着它没有从开始时移动），我们知道y不是从C开始的。

最后，如果玩家宣布四张牌组，我们需要知道该组中的所有牌要么已经移动并被他们拥有（这种情况下我们一无所知），要么没有移动，然后我们知道这些牌是他们开始时拥有的。并且我们将所有牌标记为已知位置“已出”。

鉴于这些知识，我们如何检查所掌握的信息是否与某个初始手牌一致？约束条件足够小，以至于可以使用经过一定程度剪枝的暴力搜索方法。然而，这个问题也可以表示为一个匹配问题，即将玩家手中的32张牌与32个牌值进行匹配。

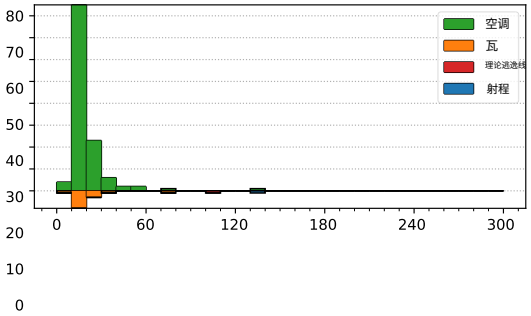
? 如果我们知道一张特定的牌是从玩家手中开始的，我们只需从该玩家手中的八张牌中选择一张，并将其与牌值进行硬匹配。
? 如果知道某张特定的牌没有出现在玩家手中，我们就删除连接玩家手中的每一张牌和该牌值的边。
? 如果我们知道玩家必须从一个未移动的集合中的一张牌开始，我们就会选择该玩家手中的某一张牌，并移除它与任何其他集合之间的所有边。

如果这个图中存在一个完美的匹配，那么可能还没有人作弊。

问题Y：压缩问题作者：Jakub Onufry Wojtaszczyk和Bob Roos由124支队伍解决。首次解决时间为6分钟。

最短的团队解决方案：305字节。

最短的法官解决方案：108字节。



这本应是竞赛中最简单的问题之一。需要注意的三个关键属性是

1.字符串的第一个字符不能改变，2.字符串的最后一

个字符不能改变，以及

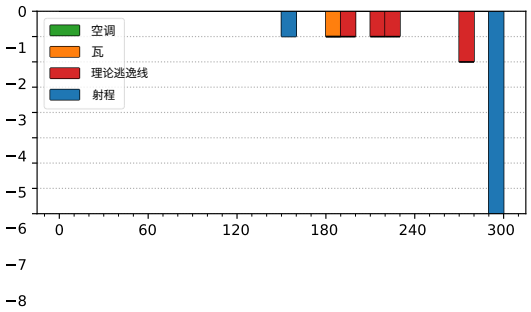
3、不可能删除一个字符的所有出现。

这样，二进制字符串的可能答案集就缩小到长度不超过3的字符串。这可以通过移除字符使所有相邻字符不同，然后移除大小为2的前缀来实现。对于所有的二进制字符串，都存在一个唯一的最短结果。

问题Z：考古恢复问题作者：

Federico Glaudo和Per Austrin

由0支队伍解决。
最短的法官解决方案：2537字节。



在评委看来，这是这组题目中最难的一道。

每个金字塔位置对应于Z3中的一个向量，其中安克对应0，眼睛对应1，鹦鸟对应2。同样地，每个杠杆也对应于Zk3中的一个向量，不移动金字塔对应0，顺时针移动对应1，逆时针移动对应2。移动杠杆对应的向量u相当于将代表杠杆的向量加到代表金字塔的向量上。

所以，在背景故事中，我们在Zk3中有n≤40个杠杆向量。我们取了这些向量中的一些的所有可能的2n次和，在每种情况下都得到了一个描述所有金字塔位置的向量；称这些向量为（si）2i$_{=1}^{n}$我们被要求找到一些可能的向量序列（uj）nj=1，这些向量序列能够生成观察到的si。这是一个难题，我们将分步骤解决它。

第一部分：k = 1首先，让我们对一维问题做一个观察。在这种情况下，（uj）就是Z3中的一个多重集，同样地，（si）也是如此——因此（si）只是零、b个一和c个二。我们来看看向（uj）添加一个数后，多重集（si）会发生什么变化。

如果我们加上一个零，那么a、b和c就会乘以二。如果我们加上一个1，那么a、b、c就会变成a + c、b + a、c+b（加上2的结果类似）。如果我们先加上所有的1和2，我们可以看到这些数的重数（a、b、c）总是要么是{x，x + 1，x + 1}或者{x，x，x + 1}，顺序不同而已。重要的是，只要我们只添加对应于1和2的杠杆，其中至少有一个数字a、b、c总是奇数。因此，对应于零的杠杆数是a、b、c的所有元素的最大的二的幂。

第二部分：在线性子空间上求和现在，让我们尝试将k = 1的推理应用到更一般的情况。考虑任意向量x∈Zk3，并取该向量与所有向量的内积。更确切地说，考虑数列（uj · x）nj=1；那么这些数的所有可能和的多重集是（si · x）2i
$_{=1}^{n}$, because scalar products commute with addition因此，从第一部分中，我们可以找出有多少个uj满足uj · x = 0，对于任意的x。我们用f (x)表示|{j： uj · x = 0}。可能的x只有3^k种，而si的取值最多有3^k种，所以我们可以计算所有可能的x对应的f (x)，复杂度为0（k^32^k），这已经足够快了。

这意味着对于任何维度为k-1的线性子空间，我们可以知道有多少

这些ujs都在这个子空间中。我们可以用多种方法来缩小范围。例如，考虑任意维度的子空间H，以及G=H⊥，然后观察Σx∈H f (x)。这是

$$\sum_{x\in H} f(x) = \sum_{x\in H}\sum_{j=1}^{n}[x \perp u_j] = \sum_{j=1}^{n}\sum_{x\in H}[x \perp u_j] = \sum_{j=1}^{n}|\{H \cap u_j^{\perp}\}|,$$

是H和u的交集的大小$^{\perp}_{j}$, summed over j一个向量uj对这个和的贡献是多少？如果uj∈G，那么uj与H中的每个向量都正交，因此它对H的贡献为|H|。否则，H与u的交集为$^{\perp}_{j}$ will be a subspace of dimension one更小，因此uj将贡献|H|/3。所以，通过计数，对于任意G，我们可以计算出G中有多少个uj——它是

$$\frac{1}{2}(\frac{3\sum_{x\in G^{\perp}} f(x)}{|G^{\perp}|} - n).$$

在计算了任意x的f (x)之后，我们就可以用O（|G⊥|）的时间计算出任意G的上述值。

第三部分：翻转符号首先，取G = {0}—单维子空间。我们找出集合{uj }中有多少个全零杆。然后，取G = {0，x，?x}对于任意的x。我们计算出有多少个uj在G中——由于我们已经知道有多少个uj是零，因此有多少个uj在集合{x，?x}中，对于任意的x∈Zk3。对所有（3k?1）/2个x值进行计算需要O（3^k）时间。因此，我们知道所有uj以及符号的选择；现在我们需要最后一个想法。

我们可能有最多2^n种可能性来分配符号以进行检查（如果存在零向量或任何一组{x，?x}包含多于一个元素，则实际数量可能会更少）。我们需要找到一个可行的方法。请注意，如果我们计算某些u1，...，un的s1之和，那么对于u1，...，un?1，?un的s ′ i之和将是多重集{si?un}（双射关系是将包含n的所有索引映射到去掉n后的相同集合，反之亦然）。因此，我们采取以下步骤：

1. 我们为ui选择任意一组符号，并计算多重集si。
2. 在做这个的时候，对于多重集si的任何元素，也要记住以某些uis之和的形式到达它的任何一种方式。
3. 对于Zk3中的任何向量v，查看多重集{si?v}，并检查它是否等于输入中给出的多重集。
4. 对于任何有效的v，看看它是否可以表示为一些u的和，以及如何表示（这就是我们在第2点中存储的到达元素的方法）。
5. 如果我们找到了一个，那么我们就翻转用来到达v的uis的符号，保持其余不变，并输出它作为解决方案。
6. 如果导致输入多重集的vs不能表示为某些uis之和，则输出不可能。

您也可以查看本文或本文，以详细了解更通用的问题。