### 均值加变乘例子

在均值加变乘大动作中，给出各种题目，其将通过穷举小动作，自我进化出好的算法。

1. Quest1: a+1/a2>3，连接到divideN算法。

另见，[增减项改值算法](#增减项改值算法)

1. 套用公式 x+y≥2√(xy)有: a+1/a2≥2√(a/a2)
2. 此前，当**事先撰写n等分动作divideN**。eg. a=a/2+a/2
3. 需求入参的表达：n倍（做为乘项）a的指数

调用方可传入公式来表示：你方对表达式进行处理，要求处理后的结果经由我方提供的公式来匹配替换，替换后可或n倍其指数（或达到高于rate的化简比)。

1. 自动分析出功能特征：n等分后的加变乘，将n倍其指数。

这是本动作反复验证后基本发现的规律 。

1. 挖掘期望特征。

基于可抵消形式的篡改有助于提出期望特征。

对于 a+1/a2≥2√(a/a2)，若可2倍分子的指数，可全抵消。

可在成功失败步骤的差异中寻找期望特征、或从成功步骤中寻找期望特征、或直接设定期望特征：可化简。

1. 优先执行【功能特征同期望特征相配】的动作。

优先执行2等分动作。

a+1/a2 = a/2+a/2+1/a2  ≥3(a2/(4a2))^1/3, 成功大抵消。

1. 根据上面的实例，构造训练数据。

自动生成类似例子，总结出频繁特征：加式有son和shuSub同底不同指，

若指数是n倍关系，则进行n等分后执行乘变加。

1. Ok.
2. u265a: a+27/(1+a)3≥3
3. 变乘后右侧不可抵消。有特征：son同chuSub的底数相似，故可以关联到动作 like2eq()。
4. 执行后发现完美关联到Quest1相关动作。

执行之。

1. 完美抵消，成功。
2. ```
3. u707b: a+1/(b(a-b))≥3

see [增减项改值算法](#增减项改值算法)

1. Quest2: a+b/(1+a)>c ~> (1+a)+b/(1+a)-1>c~>2√b-1>c

See [增减项改值算法](#增减项改值算法)

1. u2750: 3a+3b+9/(1+a2+b2)+a2+b2≥11
2. 加子聚集只是一个动作，当全面了解它。

对于9/(a+b)+a+b，有动作集：加变乘，加子聚集。

有特征：除底的子存在于根子中。

发现：若除底在若干根子s’中,可对s’做聚集，这样做最终导致加变乘后可抵消。

当把这个动作映射加入到均值加变乘的小动作树训练数据后，就可忽视其存在啦！

1. 上面聚集后式子变成了：9/(1+x)+x≥11

它匹配了[增减项改值算法](#增减项改值算法) 动作。调用之，成功抵消。

1. ```
2. t2581a:a+1/√(a-1)+√(a-1)>7/2
3. 特征：a是除底的一部分，能被优先关联到 like2eq动作。

处理后：(a-1)+1/√(a-1)+√(a-1)+1>7/2

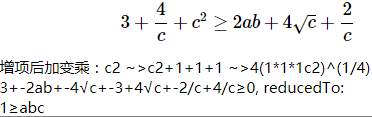
1. 变乘后发现特征：a-1在乘集中的指数和是除集中指数和的3倍。

根据Quest1中的divideN动作，需要对1/√(a-1)进行3等分。

1. 均值5项加变乘
2. t200170a AvgJ2c\_addOnesAct

设法对 c2 降次，然后同4√c抵消。

See: n等分后变乘使次数乘n，n次连加1后变乘使次数除以(n+1)。



1. 撰写n次连加1小动作，及其出入时特征（简单）

入时特征：左右侧同底炮，n倍次数差。

出时特征：可n倍降次。dropPowSup via avgJ2c

功能特征：根据入时特征n倍降次: 连加n-1个1即可。

1. 通过穷举，学会：在均值乘变加前，执行该小动作即可。
2. ```

-----------------

1. ✱✱ 基于公式自动改值后，使用【增减项改值算法】使替换侧变得可化简。
2. 公式x+y≥2√(xy) 有实例 a+b/a ≥2√(ab/a)

且 有映射x~>a, y~>b/a, 即y中有x

1. 对y中的某个x实例改值：b/a~>b/(a+1),

于是有 a+b/(a+1) ≥2√(ab/(a+1)) ；#2

因匹配公式，它仍是真式，但丧失了抵消性。

这引出了一个需求特征：期待抵消。

1. 统一改值使可匹配且可抵消：对其余x统一改值

(a+1)+b/(a+1)-1 ≥2√((a+1)b/(a+1))

1. 以上几步演示了：

#2式左侧虽可被匹配替换，但替换式无法化简

于是可考虑先行改值，使替换式可化简。

似乎类似改值需要一个统一的，但并不复杂的【增减项改值算法】。这使得我们在公式匹配替换时，若被替换侧不能化简，当优先尝试用【增减项改值算法】对其处理。因为#2 是可以被匹配的，没必要据此生成数据集来进行动作引导。

1. ----------------------------

只要y中有x, 也可以尝试基于步骤集中的推导式来构造数据集。

**但目前步骤集中的真式看起来也是较简的，或必要整的更复杂？**

通过对推导式改值来增加训练数据的方法如下：

1. 步骤f1(x,y) => f2(x,y) 显然是真式（原真式），有实例 f1(a,b)=>f2(a,b)。要求： 实例前件f1(具有化简特征等), 且b中有a.
2. 对所有b下的某字母a改值，使f1变得不可化简。但实例仍是真式。
3. 对实例中的其它a统一改值，要求改值后f1可化简，且要求实例仍是真式。此时f2(a,b)可能进行了增减项，只是部分匹配f2(x,y)了。
4. 只要满足大小值替换规则，此时实例当可使用原真式的处理逻辑（动作）

eg. 若原f2式是: x+y3>3x2-y #i1, 改值后变成 x+y3+1>3x2-y #i2, 那么， #i1可以成为#i2的充分条件 。

1. ``
2. ``
3. ``
4. 通过简单的加减项生成每个步骤的若干后件，以增加数据集。

这可用来应对简单的增减项攻击（改值攻击）。意义不大。

eg. a>b有后件 a-2>b, 于是在可把 a-2>b ~>addSubItemAct 加入数据集。addSubItemAct是加减项动作，用来在逆推时，试探性地加减以生成不等式的允分条件。即：欲证a-2>b，则证a>b即可。若充分条件很多，可通过在分类树中查找判断，以取得最有可能被证出的充分条件。

1. 对步骤集中的p>q <= x>y展开演算，生成大量训练数据。

它们都是真式。本质上也和对公式展开演算相似。

前提:对同名母改值后（常量|字母|算符，但不违反已知）当保证仍能解出题目。

1. 对个别字母改值。

若y中a或其某祖先A同另侧中某个部分完全对应(必等值)，则对a随机改值为a’. 完全对应是指：可用字母t换元的，且x,y侧都只有一处字母是t。

1. 对y中a的同名变量进行同步改值, 这个同步未必能成功. eg.

a+b/(a+1)>2√(ab/(a+1)) ，根据均值不等式，成功同步为：

(a+1)+b/(a+1)-1>2√((a+1)b/(a+1))

1. 对p>q中的a的同名变量进行同步改值。

只要无已知冲突，应该没问题。

1. 把改值后的p>q <= x>y加入到训练数据集。
2. ```