T1

subtask1

暴力dfs决定每个珠子选不选。

subtask2

预处理出每种数字出现的次数。

暴力 dfs 决定每种数字选还是不选。

对于一种方案，求出所有数字的 gcd，乘上方案数，加入答案。

subtask3

预处理出每种数字出现的次数。

从 1 到 m 枚举 gcd，设 f_i 为选出的数的 gcd 为 i 的倍数的方案数。

对于 i，枚举它所有的倍数 j，统计出 i 的倍数的出现次数t，则 $f_i=2^t-1$。

设 g_i 为选出的数的 gcd 恰为 i 的方案数，考虑容斥，有$g_i=f_i-\sum_{i|j,i\neq j}g_j$。

最终答案为$\sum_{i=1}^{m}ig_i$，时间复杂度$O(m\log m+n)$。

T2

source: HDU 6232

题意

数轴上有两个人，告诉你每个人的指令，即向左/右/不动维持多久，可以随意选择出发点，使得两人在整点位置的见面次数最多。

设时刻 i,a 的位置为 $A[i]=A[i-1]+R_a[i]$ ，b 的位置为 $B[i]=B[i-1]+R_b[i]$
相距 $D[i]=A[i]-B[i]$
显然，见面次数最多，就是 $D[i]$ 中出现频率最高的数的个数。
对于 $D[i]$,最多有 $2N$ 个转折点，每个转折点有可能为五种：$0,1,-1,2,-2$
针对这 5 种情况，就知道相邻指令之间经过的距离的范围了，相当于对于距离进行区间覆盖，最后前缀和，找最大的数

注意要分奇偶

因为，对于 $0,1,-1$ 距离都是连续覆盖的，但是，对于 $2,-2$ 来说，是隔一个覆盖一个距离，而题目要求在整点位置见面才算数，所以，间隔的距离是不能统计的，比如相邻指令经过的距离为 $3,5,7,9$ 显然不能对 $[3,9]$ 进行覆盖，因为 $2,4,6,8$ 不能算是出现过

分奇偶，只需要对 $1,-1$ 的情况，拆成两个进行覆盖即可。

T3

先考虑计数有多少种划分集合的方案，帮助思考：

考虑树形 DP，令 f_u 表示以 u 为根的子树的划分方案数。对 u 这个点所在点集进行分讨：

- 若点 u 所在集合的 LCA 就在点 u，那么相当于从 u 的每个儿子 v 的子树中各取**最多一个点**，将其和 u 归为同一点集。（情况 1）
- 若不在，那么这个点先搁置，将来可能会和它的一个祖先合并。（情况 2）

不难发现可以 DP 加一维，令 $f_{u,i}$ 表示以 u 为根的子树内，有 i 个搁置的点。

下面只说情况 1 的转移（情况 2 显然），新加入一颗子树 v 的时候，转移如下：

- $f_{u,i} \times f_{v,j} \to f'_{u,i+j}$（$v$ 子树内不选点和 u 匹配）
- $f_{u,i} \times f_{v,j} \times j \to f'_{u,i+j-1}$（$v$ 子树内选一个点和 u 匹配）

现在回到原题。那一坨关于 $\prod(\sum a_x)$ 的贡献，可以先乘法分配律展开，再考虑一下组合意义，就变成了：

> 先钦定一种划分方式，再在每个划分出来的集合里钦定一个点，将所有钦定点的点权乘起来，贡献到答案里。

更清晰地描述它：

> 先钦定一种划分方式，再给每个点黑白染色，每个划分出来的集合里恰好有一个黑点，将所有黑点的点权乘起来，贡献到答案里。

（我们不一定要先划分再染色，而是在 DP 的过程中，同步考虑**划分**和**染色**两件事情。也就是考虑子树 u 时，u 子树内每个点是黑是白也钦定好了。）

在上面弱化版问题的树形 DP 做法上稍加改动，下面只说情况 1（情况 2 仍然简单），有以下两种：

- 令 u 为黑点，则在每个 v 里选最多一个白点和它匹配。
- 令 u 为白点，则在每个 v 最多选一个点和它匹配（这之中恰有一个黑点，其余白点）。

那么只需要在状态里加一维：令 $f_{u,i,j}$ 表示子树内有 i 个搁置的白点，j 个搁置的黑点。

这就可以记录下转移所需要用到的所有信息了。时间复杂度 $\mathcal{O}(n^4)$。

T4

source: wf 2024 e

subtask 1

不知道指数级爆搜+剪枝能搜过不。

subtask 2

可以发现底面一共只有6种情况，然后把底面的左下角坐标和底面的形状一起压成状态，就可以bfs了。虽然是在一个无限大的平面上操作，但是由于$a,b,c,|x|,|y| \leq 100$，可以人为设置一个阈值（例如1000），如果翻转过程中坐标绝对值超过该阈值就不优了。这样总的状态数还是能控制住的。

subtask3

看x, y是否为a的倍数，然后直接输出答案。

subtask6

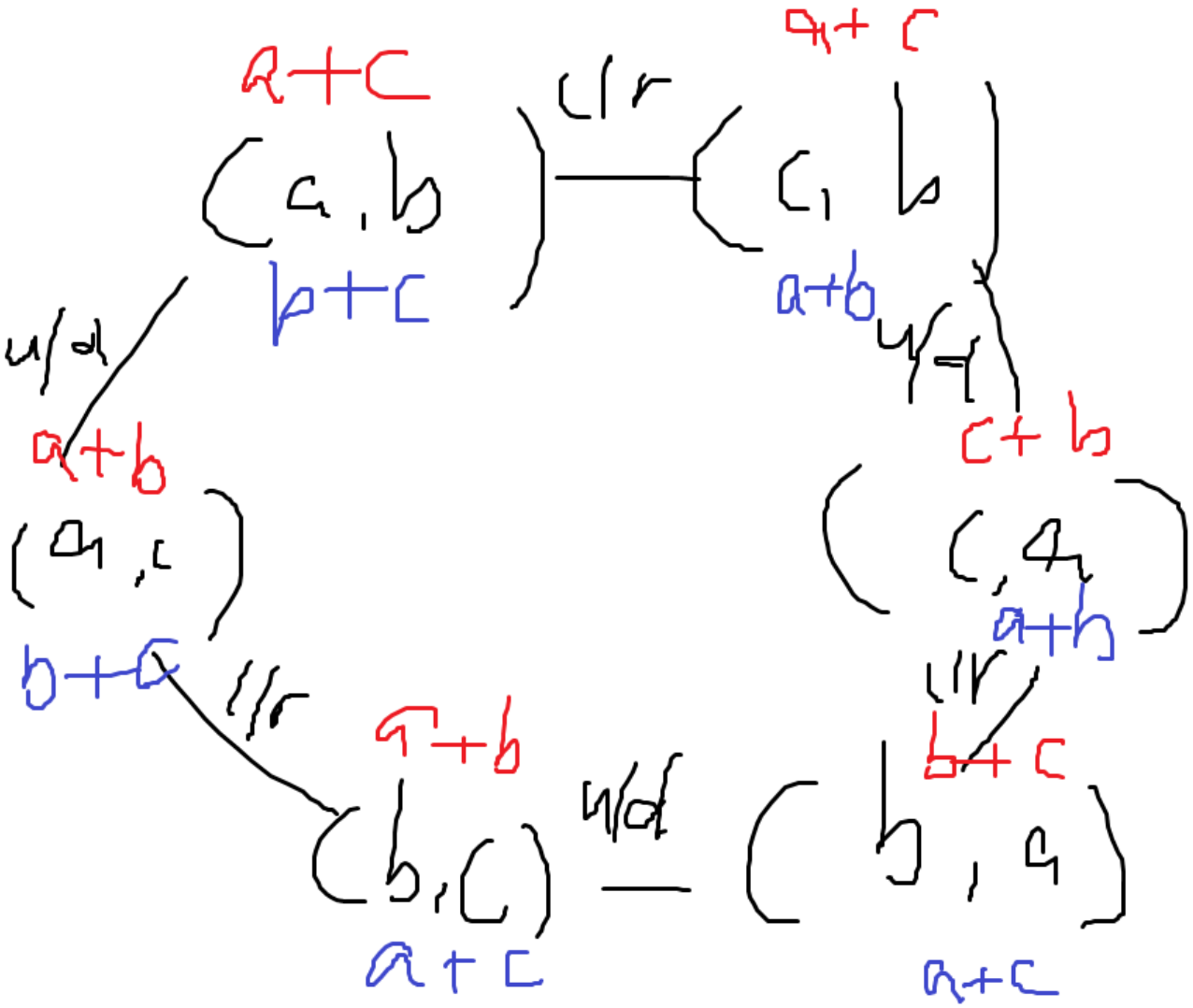
先介绍正解，然后如果你在正解过程中遇到困难，可能可以转而去做subtask4或者5。

首先我们把一次上下左右的翻转记作UDLR。我们发现执行两次连续的翻转操作后，底面的形状不变，而长方体的坐标发生了变化。具体地，如果现在底面是$(0,0)-(a,b)$，那么执行RR/LL会让横坐标增加/减少$a+c$，执行UU/DD会让纵坐标增加/减少$b+c$。通过反复执行上述操作，我们可以到达所有形如(k_1(a+c),k_2(b+c))的位置。

进一步地，我们发现在执行了一次翻转操作后，底面的形状发生了变化，具体地，执行一次R/L操作会交换a, c的地位，而执行一次U/D操作会交换b, c的地位。

我们可以想到，如果在两次RR操作之间，插入连续的翻转操作，例如RDDR，那么此时执行的DD就不再是让纵坐标减少$b+c$了，因为执行R操作会交换a, c的地位，此时的DD是让纵坐标减少$a+b$。当然，要让DD操作减少纵坐标$b+c$，仍然是可行的，只需执行RRDD这样的操作就可以了。也就是说，只要我们进行了一系列操作（例如这里的RR）后，底面形状不变，我们就不用考虑后续插入的两次连续的翻转操作具体被放在什么位置，我们只用关心两次连续的翻转操作能对坐标产生怎样的影响就可以了。例如DD就既能让纵坐标减少$a+b$，也可以让纵坐标减少$b+c$。

当然，并不只有两次连续的翻转操作让底面的形状不变。例如LURDLU，这样的操作也让底面的形状不变。操作、底面状态变化以及在该底面状态下执行连续两次翻转操作带来的x, y坐标增减如下图所示：



其中，红色表示该状态下x坐标的增减，蓝色表示该状态下y坐标的增减，黑色表示状态以及操作带来的状态转换。

因此，我们的算法会首先搜索出一些操作序列，这些操作序列使得底面形状不变。操作序列的长度最多为8，其实就是在上图上面找一条回路。至于为什么序列长至多为8，例如我找一条LURDLU也是回路，长度为6，且经过了所有状态（横纵坐标都可以增减$a+b, a+c, b+c$）。考虑URURRURU这个序列，也是回路，虽然没有经过所有状态，但是横纵坐标都可以增减$a+b, a+c, b+c$，可能你会觉得URURRURU不是有两个连续的R么，是不是可以在URUURU序列中间插入RR，就不用搜了，但是URUURU这个序列纵坐标不能增减$a+b$，和URURRURU是不一样的。那么为什么序列长不能是10，可以发现序列长如果是10的话，例如URURUURURU，这下就可以把中间两个连续的U去掉了。所以序列长度至多是8。

然后对于每个操作序列，我们根据翻转过程中经过的各种底面形状，判断在操作序列中插入两次连续的翻转操作能不能让横纵坐标增减$a+b \ or \ a+c \ or \ b+c$。假设横纵坐标都可以增减$a+b, a+c, b+c$，然后问题就变成：

找一组k_1, k_2, k_3，使得$k_1(a+b)+k_2(a+c)+k_3(b+c)=x'$，且最小化$|k_1|+|k_2|+|k_3|$（纵坐标也一样，横纵坐标独立计算）。

subtask6范围下的该问题我们先放在这里。

subtask4

有可能翻转过程中经过的各种底面形状有限，例如横坐标只能增减$a+b$，那就只用判断x'是否为$a+b$的倍数就可以了，这很简单。

稍微难一点，如果横坐标只能增减$a+b, a+c$，那么问题变为：

找一组k_1, k_2，使得$k_1(a+b)+k_2(a+c)=x'$，且最小化$|k_1|+|k_2|$

这个大家就很熟悉了，这不就是二元一次不定方程么，原题：poj2142。

在subtask4中，$a=b$，那么$a+c=b+c$，那么subtask6留的那个问题也可以从三元变成二元的，就好做了。

subtask5

在subtask5中，x, y的范围不算大，那么我们可以建图，点i向$i+(a+b), i-(a+b), i+(a+c), i-(a+c), i+(b+c), i-(b+c)$连边，边权为$1$，最短路可以bfs。当然还有个问题是范围，考虑到$a, b, c$范围更小，可以人为设置一个阈值，如果操作过程中绝对值超过该阈值就不优了。这样总的点数还是能控制住的。

由于对于不同的操作序列，a, b, c是不变的，只有x'在变化，所以我们可以把dis数组存下来，不用对每个操作序列都重新算一遍这个问题的答案。

subtask6

感觉这个东西比较像同余最短路，不妨设$A=a+b, B=a+c, C=b+c$且$A>B>C$。不过同余最短路求的是$k_1A+k_2B+k_3C \bmod A=i$时，$k_1A+k_2B+k_3C$的最小值（即dis_i）。和我们这个还是区别比较大。

不过我们还是可以用同余最短路的思想，现在dis_i不再是一个值，而是一个vector，存储每次$k_1A+k_2B+k_3C \bmod A=i$时，$k_1A+k_2B+k_3C$和$|k_1|+|k_2|+|k_3|$的值。

再用subtask5中的bfs方法，一开始将0加入队列，然后每次把i取出队列，把$i+B, i-B, i+C, i-C$加入队列，并且把i以及凑出i所需的步数加入$dis_{i \bmod A}$这个vector中。

当然这样会无限加入元素。我们先想想假设没有无限加入，应该如何对x'求答案?

就是找到$dis_{x' \bmod A}$这个vector，取出vector中的所有元素，答案为$\frac{|x'-(k_1A+k_2B+k_3C)|}{A}+|k_1|+|k_2|+|k_3|$的最小值。

那么我们也可以用这个来剪枝，当把i取出队列时，我们先看一下i对应的答案，如果凑出i所需步数大于等于目前i对应的答案，那么把i以及凑出i所需的步数加入dis就不优了，因为用现成的就可以凑，那么就不加入vector，也不把$i+B, i-B, i+C, i-C$加入队列。

由于对于不同的操作序列，a, b, c是不变的，只有x'在变化，所以我们可以把dis存下来，不用对每个操作序列都重新算一遍这个问题的答案。

考虑复杂度，由于有剪枝，我们发现至多加减$A-1$个B和C，也就是至多把这么多东西加入队列和vector。那么复杂度不会很高。