PROBLEM NAME:Button
DIFFICULTY : EASY

Let xi be the number of times the i-th element is incremented. It is easy to see that x1 xN must hold. On the other hand, when this inequality holds, it is always possible to nd such a way to press buttons (specifically, press the i-th button xi - xi+1 times). The total number of button pressings is x1.
Thus, we want to minimize x1 under the constraints that:
For each i, Ai + xi is a multiple of Bi.
x1 >= .. >= xN>= 0
In the optimal solution, we can assume that xN < BN: otherwise we can replace xN by xN - BN and the condition still holds. Thus, xN should be the minimum value that satisfies the
condition xN >=0 and AN + xN is a multiple of BN. Similarly, we can determine the values of xN - 1,... , x1 greedily in this order.
The time complexity is O(N).

PROBLEM NAME: Too easy
DIFFICULTY: HARD
PREREQUISITES : FFT, Combinatorial deductions.
CODE:https://www.codechef.com/viewsolution/27290018

First, solve the problem for a fixed value of K.
For each edge e, count the number of ways to choose K vertices such that the edge e is included in the subtree. If we compute the sum of these values for all edges, we can compute the sum of number of edges in all subtrees defined by K vertices. Since the number of vertices in a tree is the number of edges plus one, the answer is this sum plus the total number of ways to choose K vertices from the N vertices.
Assume that if we cut the given tree by the edge e, we get two subtrees of the sizes A and N − A. Then, the number of ways to choose K vertices
That include this edge is C(N,K)-C(A,K)-C(N-A,K).Thus, for each we can compute this value ,and the answer is sum of these values plus C(N,K). This can be done in O(N).
We can simplify this solution a bit. First, compute the frequency list of the sizes of subtrees obtained by cutting a single edge. We can convert it to a sequence of coefficients b0 , b1 , . . . , bN (bi = N if i = N , otherwise bi is minus the number of subtrees of size i), and the answer is simply the following:
∑ bi*C(i,K)
Let’s compute this value for all K efficiently. By using C(i,K) = i!/(K! * (i-K)!).

∑ bi*C(i,K) = 1/K! *∑ (bi*i!)*1/(i-K)!

Let ci = bi × i! and di = 1/(−i)!. Then, this value can be computed
by the sum of ci × dK−i . This can be seen as a convolution, thus we can
compute these values for all K using FFT in O(N logN).

PROBLEM NAME:Cuboidaloid
DIFFICULTY : MEDIUM
Code : https://pastebin.com/vYG4gEts

The 2D version of the problem is easy. Consider two adjacent rectangles. If these two rectangles are
not ”aligned” well, we can uniquely determine the positions of more rectangles, and we will eventually
fill a entire row or a column. Thus, there is a line parallel to one of coordinate axis that doesn’t split
any rectangles. It’s not hard to count such patterns.
Now, let’s solve the original 3D problem. Assume that A, B, C are multiples of a, b, c, respectively
(otherwise the answer is zero). We call a pattern ”trivial” if there is a plane that doesn’t split any
cuboids. We can count the number of trivial patterns easily. The number of patterns that can be cut
by a plane can be reduced to the 2D case. Do not forget to avoid double-counting by using
inclusion-exclusion principle. For example, you should subtract the number of patterns that can be cut by planes
of two directions.
The main challenge is that, in 3D case, there are non-trivial patterns. Let’s think how these patterns
look like.
First, for a torus cuboid with parameters (p, q, r) in the statement, write an integer k into the
small-cube {((p + i) mod A, (q + j) mod B, (r + k) mod C))}. This way all small-cubes will
contain an integer. Let v(x, y, z) be the integer written on (x, y, z) For each pair (x, y), the sequence
v(x, y, 0), v(x, y, 1), · · · , v(x, y, C − 1) will be a cyclic shift of 0, 1, .., c − 1, 0, 1, .., c − 1, .., 0, 1, .., c − 1.
Thus, the values of v(x, y, 0) determines all values of v. Let v(x, y) = v(x, y, 0), and consider an A × B
table whose (i, j)-element is v(i, j).

Now, for a given table, we want to count the number of patterns that are consistent with this table.
Fix a pattern that is consistent with the table. In each layer (here a layer means a plane with constant
value of z-coordinate), we get a partition of the entire $A \times B$ rectangle into $a \times b$ torus rectangles that
corresponds to the pattern. Here, notice that a torus rectangle never contains two cells with different
values of v, and the partition at the layer with $v(x, y, z) = 0$ determines everything else.
Thus, we get the following. Let $f(h)$ be the number of ways to partition the set of cells (x, y) such
that $z(x, y) = h$ into torus rectangles. Then, the number of patterns that is consistent with the table is
f (0)^C/c × · · · × f (c − 1)^C/c .
Now, fix an $A \times B$ table (with the values of v). When do we have non-zero number of non-trivial
patterns that is consistent with this table?
We call this table ”row-aligned” if in each row, all numbers are the same. Similarly, define
”column aligned”. If the table is both row-aligned and column-aligned, it means that the table only contains a
certain constant, and this corresponds to patterns that can be cut by xy-plane. Since this pattern is
trivial, we can ignore it. If the table is row-aligned (or similarly, column-aligned), the table is multiple
stripes whose heights are multiples of a (see the picture below). In this case the only way to partition
it into torus rectangles is to entirely divide it into stripes with heights a, thus again it corresponds to a
trivial pattern. Thus, we assume that this table is not aligned in any directions.
Consider a way to partition this table into torus rectangles of dimensions $a \times b$. Each torus rectangle
must contain the same values of v. As we see in the 2D case, this partition is either ”horizontal” (i.e., a
union of stripes of height h, some stripes are possibly shifted horizontally), or ”vertical”. If all such ways
are ”horizontal” (or ”vertical”), it corresponds to a trivial pattern. Thus, there must be both horizontal
ways and vertical ways to partition it. It means that the entire $A \times B$ table must be splitted into $a \times b$
torus rectangles in the most natural way (i.e., all rectangles are aligned like a grid). Also, since the table
is not aligned, there is a unique way to do so. Let’s call it ”standard partition”.
In order for the pattern to be non-trivial, in at least one layer the partition must be shifted to horizontal
direction, and in at least one layer the partition must be shifted to vertical direction. This means that

there exists an integer h that dominates some rows and some columns, as in the picture below:
https://imgur.com/JbCNVN8
IMG:https://imgur.com/cEEJxkC

Since we have a standard partition, now we can consider the entire table as an A/a × B/b table. As
we see above, there are h, p, q such that h dominates exactly p rows (0 < p < A/a) and q columns
(0 < q < B/b). In this case, the number of standard partition is 1, the number of horizontally shifted
partitions is b^p − 1, and the number of vertically shifted partitions is aq − 1. We want to count the
number of ways to choose a sequence of C/c partitions such that at least one is horizontally shifted, and
at least one is vertically shifted. This value is (b^p + a^q − 1)^C/c − (b^p)^C/c − (a^q)^C/c + 1.To summarize, the solution is as follows:
• Count the number of trivial patterns by inclusion-exclusion.
• Let's count the number of non-trivial patterns. First we fix a standard partition (ab ways) and
the value of h (c ways). Then, for each pair (p, q), compute the following values:
(b^p + a^q - 1)^C/c - (b^p)^C/c - (a^q)^C/c + 1.

Suppose that we have an A/a × B/b. How many ways are there to fill this tables with integers between 0 and c − 1, such that exactly p rows are dominated by h and exactly q columns are dominated by h? This can be done by a simple O(N ^4) DP ("exclusion principle").
– Compute the value (b^p + a^q − 1)^C/c − (b^p)^C/c − (a^q)^C/c + 1.
We compute the product of two values above, compute the sum for all pairs (p, q), and multiply
it by a factor of abc.

PROBLEM NAME:SIMPLESET
DIFFICULTY : EASY-MEDIUM

Define DPX[i][j] as the number of ways to divide the first i integers into two sets such that:
The last (i-th) element is in X.
The last element that is in Y is the j-th integer.
Similarly, define DPY [i][j].
What are transitions from DPX[i][j]? First, when Si+1 >= Si A, we can try to include Si+1 into the set X. In this case, for each j, we should add DPX[i][j] to DPX[i + 1][j]. Also, when Si+1 -Sj>= B, we can try to include Si+1 into the set Y . For each j that satisfies the inequality

above, we should add DPX[i][j] to DPY [i + 1][i].
Now, instead of creating a 2D array, we keep an array DPX[j] that represents DPX[i][j] in the definition above for xed i. What happens when we increment i? First, compute the sum of DPX[j] for all j that satisfies the condition Si+1 - Sj B, and remember this value. Then, if Si+1 - Si < A, "clear" the DPX array. Then, add the remembered value to DPY [i]. Do a similar thing for DPY array.
Thus we need a data structure that supports the following:
Compute the sum of numbers in the given range.
Update the value of one element.
Clear the data structure.
This can be done by Binary Indexed Tree and the set of positions that are updated. When you
want to clear the data structure, you write zero to all updated positions one by one. This way, when you are given O(Q) update queries, a single clear query may be slow, but the amortized
complexity will be O(Q).
The total complexity of this solution is O(NlogN). Also, if you use two-pointer method, you can get an O(N) solution (but a bit complicated).

PROBLEM NAME:sherlock
DIFFICULTY:MED-HARD
Code:https://www.codechef.com/viewsolution/27286563

let the prime m = p . Then
Trying all integers from 1 to x is too slow to solve this problem. So we need to find out some features of that given equation.

Because we have a^p−1≡1(modp) when p is a prime, it is obvious that a^zmodp falls into a loop and the looping section is p−1. Also, zmodp has a looping section p. We can try to list a chart to show what n⋅a^n is with some specific i,j. (In the chart shown below, n is equal to i⋅(p−1)+j)

chart :

Proof for the chart shown above: For a certain i,j, we can see that
n⋅a^n mod p=((i⋅(p−1)+j)mod p)⋅a(i⋅(p−1)+j)mod(p−1)=(j−i)⋅a^j mod p
And it's not hard for us to prove that n⋅a^n mod p has a looping section p(p−1). So we don't need to list i≥p.

Therefore, we can enumerate j from 1 to p−1 and calculate b⋅a−j. Let's say the result is y, then we have j−i≡y(mod p) (You can refer to the chart shown above to see if it is). So for a certain j, the possible i can only be (j−y),p+(j−y),…,p⋅t+(j−y). Then we can calculate how many possible answers n in this situation (i.e. decide the minimum and maximum possible t using the given lower bound 1 and upper bound x). Finally we add them together and get the answer.

Time complexity: O(p log p) or O(p) depending on how you calculate y=b⋅a−j.
By the way, you can also try Chinese Reminder Theorem to solve this problem.