Code Jam 2021 - Round 1B

Analysis: Subtransmutation

To solve this problem, we can first tackle a simplified version: given a unit of metal x, can it be used to produce all the metals required by the input?

To solve this simplified problem, we can use a greedy strategy. Keep track of the multiset H of units of metal we have (starting with $\{x\}$) and the multiset D of units of metal we owe (starting with the ones given as input). Iterate the following steps:

1. Remove the intersection $H \cap D$ from both H and D.
2. If D is empty, the answer is yes.
3. If H is empty and D is not, the answer is no.
4. Take all c units of the metal i with the largest number still in H. Remove all c units of i from H and insert c units of $i - \mathbf{A}$ and c units of $i - \mathbf{B}$ into H (if either new metal number is invalid, skip those).

This procedure works because those units that we greedily transform are not something that we owe, so we have nothing better to do with them (it may be futile to convert them, but it does not hurt). And since we have to produce the units that we owe, we might as well just pay them back as soon as possible, as anything we could do with the current unit we could also do with a future unit that we would use to pay the debt instead.

Test Set 1

In Test Set 1, notice that if it is possible to find an answer m for an input where $\mathbf{N}$ and all $\mathbf{U_i}$ have maximum value, that procedure also produces enough units for any other possible input. If we implement the greedy strategy above and try it for this particular input on increasingly large values for m, we can quickly arrive at the realization that $m = 29$ solves it. This means that all inputs can be generated with a metal with number no greater than 29. So, using the same procedure of checking increasingly large values of m for the given input is a valid solution for the full Test Set.

It is also possible to notice and/or prove theoretically that there are no impossible cases and that the answers are small. A quick argument is that numbers grow exponentially: from a single unit of number m we can create 2 units of number $m - 2$ (with leftovers). This means we can create 2^i units of element $m - 2i$ from a single unit of element m. We can distribute these among the elements $m - 2i - \mathbf{N} + 1$ through $m - 2i$ (with a lot of leftovers), getting $2^i / \mathbf{N}$ units of each of $\mathbf{N}$ consecutive elements. These can be further moved to elements with lower numbers if needed. This shows that an m such that $2^{m-2\times20}/20 > 20$ ought to be enough, and $m = 49$ fulfills such a requirement.

Test Set 2

As the samples show, impossible cases can happen in Test Set 2. This means running our Test Set 1 solution as is will result in the program not finishing and getting a Time Limit Exceeded error. A simple solution, however, is to notice that there is also a somewhat small maximum answer for all possible cases, and then simply stop once we are past it and return impossible.

Knowing what that maximum answer is, and proving it, is harder as we cannot just rely on experiments. We have to lean more on the theory of the problem.

We can first notice that starting from metal m, every metal we can possibly produce has the same remainder as m modulo the greatest common divisor between $\mathbf{A}$ and $\mathbf{B}$ (let us call that g). This is an application of Diophantine equations, but while knowing that specific theory could help tackle the problem, it is not absolutely necessary. Now, consider the remainder modulo g of every i such that $\mathbf{U_i} > 0$. If they are all the same value k, then m modulo g needs to be k as well. If they are not all the same, then the case is impossible.

At this point, we can express all metal numbers that matter as $g \times x + k$ for some x. To show that all of remaining cases are possible, we can generalize the proof we did for Test Set 1 and use Bézout's identity to build them, or we can simply iterate all relatively prime values for $\mathbf{A}/g$ and $\mathbf{B}/g$ and solve the largest case experimentally. If we do that, we can arrive at the conclusion that the maximum possible answer for a possible Test Set 2 case is 402. Therefore, we can do the same solution as in Test Set 1, exploring values for m up to 402, and returning impossible if we did not find an answer.