Code Jam 2019 - Round 2

Analysis: Contransmutation

Test Set 1

In Test Set 1 everything is small. One possible solution is to simulate the process. There is never a reason to not convert elements other than lead, so we should just apply the transformations over and over until no more lead can be created. If after a long time the amount of lead keeps growing, that means the total is unbounded. In the Test Set 2 section we do a formal analysis on bounds, but for this one, using an intuitive definition of "a lot" should suffice. If the amount is bounded, 1 gram of a metal can transform into at most 512 grams of lead, since it can only go through 9 splits before getting into a cycle. That means if we ever see more than 5120 grams of lead, the amount is unbounded.

However, the first example shows that we need to do a little more. What if lead itself can be used to get more lead? If that's possible, and we can get any lead at all, the final amount is definitely unbounded. If it is not possible, then we have no reason to ever consume lead. One way to check is to do the simulation above starting with 1 gram each of the two elements that lead is turned into. If that leads to 0 or 1 grams of lead, then lead cannot be used to get more lead. Otherwise, it can.

Test Set 2

Test Set 2 is a lot larger, so a hand-wavy analysis and implementation will not work. However, we can use the same idea above more carefully. Since each metal takes at most **M** steps to turn into lead, we only need to do **M** iterations of a simulation, where a simulation involves going over all non-lead metals and converting all grams of them. This takes $O(\mathbf{M}^2)$ time. To prevent the numbers from growing too big, we can simply do this process twice: the first pass creates as much lead as possible from non-cyclical sources, so if the second pass creates even more, then it must come from cyclical sources that ultimately yield an unbounded amount of lead. The second part, to check for lead creating more lead, is the same as before.

Notice that the result does not fit in 64 bit integers, though, and doing modulo all the time as usual doesn't work right away, since we cannot afford to equate "no X" with "10^9+7 grams of X" if X has the capability of creating unbounded amounts of lead (with X being lead itself, or some other metal). There are multiple ways to get around the problem, including using big integers, storing an additional bit for each result that represents whether it is an actual 0 value, and doing calculations modulo $10^9+7 \times K$ for some large random prime K until the very end.

There are other solutions that only work for Test Sets 1 and 2, that are less efficient implementations of the solution for Test Set 3, that is explained below.

Test Set 3

For Test Set 3, the solution looks quite different but it is actually similar. We check for cycles that may generate unbounded lead, and if there are none, we can do the simulation with a single iteration, ordering the metals appropriately. All of those things can be done in linear time, and we describe how below.

One useful thing to notice early on is that if we can determine that the amount of lead we can produce is bounded, the only formulas that are worth using are those that can eventually (perhaps in conjunction with other formulas) produce lead. Moreover, if the amount of lead is

bounded, there is no point in using lead as an input to any formula, because even in the best case, every unit of lead will turn into one unit of lead and one unit of something else that cannot be converted to lead. (Otherwise, we could simply apply that series of formulas and generate as much lead as we want).

These observations imply that we can only get an unlimited amount of lead if some metal can produce itself and lead at the same time. In other words, there must be a series of formulas that takes one unit of some metal X and turns it into one unit of X and one unit of lead. In the process, we may end up with some other metals as well, but those don't matter. Notice that if that metal X directly produces metals A and B, then we require that either A produces X and B produces lead, or vice versa. This is because even if X is lead itself and either of A or B is also lead, without considering the other result, we are just destroying 1 unit of lead to get 1 unit of lead, so the total amount of lead doesn't change. In other words, in the unbounded case, after applying each formula, we need to work on both of the outputs, not just one of them.

Let Q be a graph in which each metal is represented by a node, and there is an edge from node u to node v if, by consuming 1 unit of u, we can produce 1 unit of v. Notice that if a metal i is initially not present (that is, $\mathbf{G_i} = 0$), and it is not reachable in Q from some j with $G_j > 0$, then it is impossible to produce metal i. Therefore, we can remove any such metals at the outset. In what follows, we assume that Q contains no such metals.

Let Q' be the transpose of Q, and let L denote the node representing lead. Then, traversing Q' from node L defines a subgraph Q'_L, where every node is a metal that can produce lead after a series of transformations. Now, let P_u be the number of simple paths (paths without cycles) from L to any other node u in Q'_L. If the amount of lead is bounded, then P_u is the number of units of lead we can get from every unit of u. This follows from the fact that every (reversed) edge is a valid transformation where we start with 1 unit of some metal and produce 1 unit of some other metal, so along the path, we are using the results of previous transformations and not the very first unit of metal we started with. For example, let Q'_L be the following graph:

- $1 \to 2$
- $1 \to 3$
- $2 \to 4$
- $3 \to 4$

Which means the valid transformations are:

- 4 turns into 2 and 3
- 2 turns into 1 and something else
- 3 turns into 1 and something else

There are 2 simple paths from 1 to 4, namely, $1 \to 2 \to 4$ and $1 \to 3 \to 4$. Now, 1 unit of metal 4 can be turned into 2 units of lead (1) as follows:

- Turn one unit of 4 into one unit of 2 and one unit of 3
- Take the one unit of 3 from step 1 and turn it into one unit of 1 and one unit of some other metal
- Take the one unit of 2 from step 1 and turn it into one unit of 1 and one unit of some other metal

On the other hand, notice that the nodes that form a strongly connected component (SCC) in Q are a set of metals that can produce each other. More formally, if u and v belong to the same SCC, we can produce 1 unit of v from 1 unit of u, and vice versa. With these definitions, we can say that the amount of lead is unbounded if, for some node u in Q with v_1 and v_2 being the two outputs of node u's formula, either:

- v_1 is in Q'_L and v_2 and u are in the same SCC, or
- v_2 is in Q'_L and v_1 and u belong to the same SCC.

Otherwise, there is a limit on the amount of lead we can get, which can be computed by multiplying P_u by the initial number of units $\mathbf{G_u}$ for each u in Q'_L.

We can compute all the SCCs in O(**M**) time using Tarjan's algorithm or Kosaraju's algorithm , since both the number of nodes and the number of edges in the graph are linear on M. Walking over the graphs defined above also takes O(**M**) time, and so does checking the unbounded condition. We can also compute all P_u in O(**M**) time by taking advantage of the fact that if the amount of lead is not unbounded, then there are no cycles in Q'_L. (If a cycle existed in Q'_L, then by reversing the edges, we get at least one metal for which the unbounded condition holds.) Therefore, the nodes on Q'_L can be topologically sorted. That means we can compute a function F from vertices to the number of simple paths in linear time, using dynamic programming with this recursive definition:

- F(L) = 1
- F(u) = sum(F(v)) for each v, such that there is an edge from v to u in Q'_L

Since the number of edges in every graph described above is also O(**M**), the whole problem can be solved in linear time.