Code Jam to I/O for Women 2014 - Code Jam to I/O for Women

Analysis: Zathras

Zathras: Analysis

Small dataset

The Small dataset is an exercise in understanding the rules of the described system, and then turning them into code. There are some issues to watch out for:

- If we use floating-point numbers, we must avoid precision issues and ensure that we round down appropriately whenever the rules require it. It is safer to reframe the calculations to use only integers; in most languages, integer division already has the round-down property that the rules require.
- When determining the types of babies, we must see how many guaranteed babies of each type we get before we can know the size of the pool of remaining babies to divvy up. Suppose we have **α** = 40 and **β** = 20, and 4950 of each type of Zathrinian. Then we must create 99 babies; floor(99 * 0.4) = 39 of them must be acrobots, floor(99 * 0.2) = 19 of them must be bouncoids, and then the remaining 99 - 39 - 19 = 41 of them must be 20 acrobots and 21 bouncoids, for a total of 59 acrobots and 40 bouncoids. If we had instead tried to directly calculate the number of remaining babies as 100 - 40 - 20 = 40% of 99, we would have gotten 39 or 40 (depending on how we rounded), not 41.

Since **Y** ≤ 100 in the Small dataset, simulation is fast enough. Note that it is possible for **Y** to equal 0!

Large dataset

In the Large dataset, **Y** can be as large as 10^{15}; there is not enough time in the submission window to simulate that many steps. (Most contestants who attempted but did not pass this dataset got the "Time expired" verdict.) We need another insight.

The example in the problem statement provides a hint: the numbers of acrobots and bouncoids reach stable values that do not change from year to year. If we ever find that those numbers are the same two years in a row, we know they will be the same forever (since the rules do not change), and so we can stop the simulation. But how can we convince ourselves that this will necessarily happen fast enough, or at all? If there is a cycle of length greater than 1 — that is, if we see the same pair of acrobot and bouncoid population values more than once, but not back-to-back — then this strategy will fail. It turns out to be difficult to prove that there are no such cycles, or even to set bounds on how long the simulation can go on.

At this point, one option is to just assume that the simulation always reaches a stable point, and this assumption turns out to be correct. Another option is to implement a more cautious simulation that includes a cycle detection algorithm that uses constant memory; if we detect a cycle, we can then figure out where in the cycle the simulation would end, without actually running the remainder (but, again, this turns out not to happen). Since there are just over 10^{12} possible ordered pairs of acrobot and bouncoid population values, the simulation cannot possibly take more steps than that, and it is possible to convince yourself that much tighter upper bounds exist. Under the given rules, the percentage of acrobots approaches **α** + (100 - **α** - **β**) / 2, and the percentage of bouncoids approaches **β** + (100 - **α** - **β**) / 2, and these approaches are at more or less exponential speed, although they may oscillate around their final values for a while (and it is even possible for the total population size to increase, up to a point). The behavior of the exponential approach is "smooth" — small changes to a set of initial values do not change the behavior much — so you can make yourself more confident overall by experimenting with various values. In practice, each simulation takes at most several tens of thousands of steps.