Code Jam 2022 - World Finals

Analysis: Goose, Goose, Ducks?

Test Set 1

Consistency in this problem has a property which is not true for most typical logic systems, in that a set of statements is consistent if and only if each subset of size 2 of it is consistent. The left to right implication is trivial, but the converse is not, so let us prove it.

Assume you have a set of statements S such that any two of them are consistent. Then, for each bird b, consider all statements that state that b is at a specific point in time and space. If we sort all those points by time and linearly interpolate between consecutive points, we obtain a path for b that is consistent with all statements in S. Notice that, because any pair of statements is consistent — in particular, pairs of consecutive statements — the linear interpolation part does not exceed the maximum speed. In this way, we can obtain a path for each mentioned bird, all of which are consistent with all statements. Then, by definition, S is consistent.

Meetings also state "this bird was at this time-space point" for birds that are ducks. Thus, an analogous reasoning shows that a set of statements and known ducks is consistent with the meetings if and only if they are pairwise consistent. Since the meetings are pairwise consistent according to the limits, this leaves only pairs of two statements and pairs of a statement a meeting to check.

At this point, with the limits in Test Set 1 being so small, we can simply try everything. We know there is at least one duck, so start by trying every bird as "the first duck". Within each option, go through statements and check them against meetings (if they involve a known duck) and against previous statements. If a statement contradicts a meeting, the issuer of that statement must be a duck. If a statement contradicts a previous statement, then the issuer of such previous statement must be a duck (since ducks cannot contradict geese). Each time we find a new duck, we start checking everything again. When we go through all statements without finding any contradictions with our current set of ducks, we are done and we have a candidate set of ducks. Notice that all birds being ducks is always a valid answer. Finally, we keep the smallest set of ducks and output its size.

This solution requires $\mathbf{N}$ iterations of the outer loop, to try every possible "first duck". Checking a pair of statements or a statement and a meeting for consistency takes constant time, as it is only checking whether birds that are mentioned in both can get from one point in space-time to another, which is a simple bit of math. Thus, checking every statement against every other statement and every meeting takes $O(\mathbf{S} \times (\mathbf{S} + \mathbf{M}))$. On every iteration through statements except for one (the last one) we find at least one additional duck, so there are at most $\mathbf{N} - 1$ iterations (remember we start with an identified duck). Therefore, the overall running time is $O(\mathbf{N}^2 \times \mathbf{S} \times (\mathbf{S} + \mathbf{M}))$. With all those variables being bounded by 50, that should fit in time.

Test Set 2

In Test Set 2, we need to speed up things significantly. We can start by making use of a more refined version of our consistency observations. As you can see in the proof, we do not need to require every pair of statements or a statement and a meeting to be consistent: only those that are "consecutive".

Formally, let us call two statements s_1 and s_2 consecutive in S if they refer to times t_1 and t_2, respectively, and to bird b, and there is no other statement in S that refers to a time t_3 such that $t_1 < t_3 < t_2$ and to bird b. Similarly, let us call a statement s in S that refers to time t_1 and a meeting m at time t_2 consecutive if there is no other meeting at time t_3 such that $t_1 < t_3 < t_2$. Notice that consecutive is not a total order, because of statements referring to the same time. However, making it a total order by breaking ties arbitrarily maintains the validity of the theorem.

Then, we can say that a set S of statements and/or meetings with known ducks is consistent if and only if any consecutive pair of them is consistent. The proof is the same as the one given above.

With the observation above, if we can maintain a sorted list of meetings and goose-made statements about each bird, we can check the consistency for each statement s in logarithmic time by only checking its consecutive neighbors (at most 2 meetings and 2 statements per bird in s). This would already reduce the $O(\mathbf{S} \times (\mathbf{S} + \mathbf{M}))$ inner-most check in the Test Set 1 solution to logarithmic time, a significant improvement.

To maintain that, we can keep a tree structure that allows insertion and lookup in logarithmic time for each bird (like `set` in C++ or `TreeSet` in Java). In this way, every time we process a statement, we simply add the new information to the appropriate birds.

We can further improve by not resetting every time we find a duck. If our structure also allows removal in logarithmic time (like the examples above), we can do the following: When we discover a duck, delete all information coming from their statements. For that, we keep a list for each bird of all information they contributed. Since each piece of information is removed at most once, the overall number of removals is at most the overall number of insertions and does not affect the time complexity.

At this point we have reduced the complexity to $O(\mathbf{N} \times (\mathbf{S} + \mathbf{M}) \times F)$ where F is only logarithmic factors. The $\mathbf{N}$ comes from the external "first duck" iteration. To improve upon that, we can first notice that, if we start from the empty set of ducks and still find some, those birds must always be ducks, and would be ducks when starting from any set. Therefore, in that case, that set of ducks is the answer. If there are no forced ducks, we need to do something different, but we know there is no inconsistency between statements.

At this point, we only need to check consistency between statements and meetings. A statement being inconsistent with a meeting is equivalent to a bird b_1 saying that bird b_2 is not a duck. Therefore, if b_2 were a duck, so would b_1. We can represent the situation with a directed graph in which the edges represent these implications. If there is a path in that graph from b_1 to b_2, then there is an implication (possibly not coming directly from a single statement) that if b_1 is a duck, then so is b_2. So, if a bird is a duck, then so is every other bird in its strongly connected component (SCC). That means we want to choose an SCC that is as small as possible. Moreover, we want to choose a final component, that is, one that is not pointed to by other components. Non-final components imply other components also being made of ducks, and ultimately bringing in at least one final component. In summary, in this final case the answer is the size of the smallest final SCC, which we can find in linear time.