Code Jam 2012 - Round 1A

Analysis: Cruise Control

This is a challenging problem requiring some insights and a careful implementation, making it a really tough nut to crack!

Solving the small - simulation

Let's look at some car *A*. If there is no car that *overlaps* with *A* (that is - no car that is less than five meters ahead or behind of *A*), then it does not matter which lane *A* currently is in, as it can change lanes instantaneously with impunity. Thus, the crucial moment when we have to make a decision is when *A* overtakes some other car *B* in front of it (that is, at the moment when *A* is five meters behind *B* and getting closer), or when some other car *C* overtakes *A*.

Since all cars go at a constant speed, after any car *A* overtakes a car *B*, car *B* will never overtake *A*. This means that cars will not overtake each other very many times in the small case - the total number will be at most the number of pairs of cars, i.e. 15. Also notice that when one car overtakes another, there are only two possibilities that we need to explore: either the faster car takes the right lane and the slower car takes the left lane, or the reverse happens. If neither possibility is viable (because one of the cars is not able to take the lane we want it to take), then someone has to turn off cruise control. These two observations allow us to do a direct simulation of all possibilities.

To do this, we begin by finding out all moments in time when two cars would intercept one another, and we then look at them in order starting from the earliest. Whenever two cars meet, we check which lane they are in right now, and whether they can change to the other lane. If both cars can change lanes, we have two possibilities, and we branch out to explore them both. If one of the cars is blocked, we have only one possibility - the car which is free to switch lanes has to take the free lane, and we continue without branching. The same thing happens if both cars are blocked but are in different lanes. If the two cars are blocked in the same lane, we know someone has to turn off cruise control, and we return the current time from this branch. Finally, if we process all the overtaking events and still nobody needs to turn cruise control off, we have found a way for everybody to drive on cruise control indefinitely - we now already know the answer for the whole test case!

Since we want everybody to continue without turning off cruise control as long as possible, in the case with two branches, we should choose the branch which returned the higher value. As we branch at most 15 times, and we are able to check whether a car can change lanes simply by examining all other cars, our solution will easily run in time.

Solving the large - postponing choice

The previous strategy will obviously not cut it for the large test case. With 50 cars, there could be 1225 interceptions, and there is no way you can try 2^{1225} different possibilities! We will have to postpone making choices for as long as possible to avoid branching.

Suppose we have two cars: *A* and *B*, and *A* overtakes *B* at some point. They now drive side by side with *A* gaining on *B* over time. One of them is occupying the right lane and one is occupying the left lane, but we do not know which is which. If *A* manages to move a full five meters ahead of *B*, it is again free to change lanes, and the choice - which side it passed *B* on - is irrelevant.

On the other hand, let's see what happens when a third car, *C*, comes along and tries to overtake whoever is in the back (let's say it's still *A*); moreover assume that *C* is driving in the right lane, and cannot switch to the left. This means that if *A* is in the right lane, we will have to turn cruise control off now; on the other hand if *A* is in the left lane, we will be able to drive on for a while. This means that putting *A* in the left lane was a strictly better choice.

This leads to the idea of postponed choices. Although the choice of which lane *A* takes had to be made some time ago, it becomes relevant only now - so let's say we reveal our choice only now. Before *C* came along, we think of *A* and *B* as being in an indeterminate state, with either car possibly being in the left lane, and the exact choice is forced on us only with the arrival of *C* (one of the Googlers working on this problem said it reminded him strongly of Schroedinger's cat).

Solving the large - undetermined lanes

To formalise this approach, we will say that at any given moment of time, a car either is in a fixed lane or in an undetermined lane. For instance, in the situation described before, car *C* was fixed in the right lane, while cars *A* and *B* were initially in undetermined lanes. When *C* overtook *A*, the lanes of *A* and *B* became fixed (to the left and right lane, respectively).

Notice that we also had additional information - although *A* and *B* were in undetermined lanes, we knew that they were nonetheless in different lanes. In fact, the state of the whole system can be described at any given time by the following information:

- for each car, whether it is fixed in the right lane, fixed in the left lane, or in an undetermined lane,
- for each pair of cars in undetermined lanes, whether they are necessarily in the same lane, in different lanes, or whether they are independent of each other.

At this point, you might wonder how two undetermined-lane cars could be forced to be in the same lane. The way this happens is that there is a third car (also undetermined) that blocks them both from changing lanes. If you examine the fourth sample case thoroughly, you will find that this is exactly what happens at the twelfth second. The cars with speeds 2 and 4 are in undetermined lanes but they are necessarily the same lane, and so one of them has to turn off cruise control.

The initial state of the system is easy to calculate. Any car that is initially adjacent to another one is in a fixed lane (the lane it starts in). Any other car is in an undetermined lane, and independent from all others. The tricky question is how to update this state over time.

Solving the large - updating the state

The state changes in two situations - when two cars get close to each other (and their states stop being independent), and when two cars stop being close to each other. We can calculate all these events up front and order them by time, just as in the small case. This takes O(N^2 log N) time.

When two cars become close, they become *interdependent* - they have to be in different lanes. If one of them has a fixed lane, the other one now also has its lane fixed; if both were undetermined and independent, they are now still undetermined, but they have to be in different lanes. In other cases either nothing happens - like when one of the cars was fixed in the right lane, and the other one was fixed in the left lane; or someone has to turn off cruise control and we have solved the problem - like when both cars are forced in the left lane, or both are undetermined but they are forced to be in the same lane.

Moreover, when an undetermined car becomes fixed lane, it impacts all the other cars that are dependent on it - they also become fixed lane. Similarly, if two independent cars *A* and *B* become interdependent and we had a car *C* in the same lane as *A* and a car *D* in a different lane from *B*, we gain the information that *C* and *D* must be in the same lane. More generally, whenever we gain information due to a pair of cars *A* and *B* becoming close, we have to update information about any other pair of cars *C* and *D* with *C* dependent on *A* and *D* dependent on *B*. Fortunately, updating the state is very straightforward! A nice trick is to use +1 and -1 to denote the left and right lanes; and to use -1 to mean two cars are in a different lane, and +1 to mean they are in the same lane (and 0 to mean "undetermined" and "independent"). Then if, for example, we learn that *A* is in the right lane (*A = -1*), and we know that *A* and *B* are in different lanes (*AB = -1*), then *B* is in lane *A * AB = 1* - the left lane. Try this out and see how it works!

What happens when two cars stop being close? Well, if neither of them can change lanes (due to some other adjacent cars), nothing happens. They remain dependent, just as they were. The only moment when something does change is when a car is free to change lanes - its state immediately becomes undetermined and independent from all other cars.

Solving the large - putting it together

So let's see how the solution will work. We first determine the list of events (two cars becoming close or becoming far away), ordered by time. We also determine the initial state of each car (initially each car is either fixed-lane, or undetermined and independent from all others). We keep the state of each car in an array, and the dependency information for each pair of cars in a two-dimensional array.

For each event when two cars become close, we check if they are able to take opposite lanes (that is - they are not both fixed to the same lane, and they are not dependent to be in the same lane). If yes, we update their dependency (and possibly lane-fixedness), and update the dependencies between all their dependents). This takes O(N^2) time.

For each event when two cars stop being close, we check whether either of them can change lanes freely. If yes, we mark that car as undetermined and independent from all others. This takes O(N) time.

As we have at most O(N^2) events to process, the whole solution will run in O(N^4) time, which is fast enough for the limit of 50 we have on N.

Solving the large - optimizing

It's not hard to see that the solution we have could be optimized. To do this, let's notice that instead of keeping (and updating) the full dependency matrix, we can think in terms of groups of cars, since if *A* and *B* are dependent and *B* and *C* are dependent, then *A* and *C* are dependent as well. The exact nature of the dependency can be deduced from the other two dependencies. Thus, we need only keep one representative from each group of co-dependent cars, and for each car in the group remember whether it is in the same lane as the representative, or in a different lane. When two groups merge, we can merge them in O(N) time now: first we check whether the representatives are in the same lane, and then we switch everybody in one of the groups to use the representative from the other group. This makes our solution run in O(N^3) time. For extra credit, you might want to consider how to make the solution run in O(N^2 log N).

Fun fact: This problem was conceived when the author was driving on US interstate highways, and was annoyed by having to turn off cruise control frequently due to sub-optimal choices by other drivers who were unable to solve this problem.