**Code Jam 2015 - Round 3**

# Analysis: Runaway Quail

The optimal strategy will consist of running either to the left or right until we catch a particular quail, changing direction, then running until we catch a particular quail on the other side, changing direction again, etc., until we have caught every quail. Changing direction at a time other than when catching a quail would be wasteful.

Consider a partial solution where we have caught some of the quail on each side, and returned to the origin. For each side, let Q be the fastest quail we have not caught. (To simplify the analysis, if there is more than one quail on the same side with the same speed, we ignore all but the farthest one.) Each other quail R on that side must fall into one of these categories:

- R is faster than Q. In this case, R has already been caught.
- R is slower than Q, and is farther away. R cannot have been caught in this case, because we would have had to run past Q to get to R.
- R is slower than Q, and is not farther away. In this case, we might have already caught R, or we might not. But it doesn't affect our solution — if we have not yet caught R, we will run past R when we go to catch Q anyway.

So we can solve this problem with dynamic programming, where the states only need to include the identity of the fastest uncaught quail in each direction. We will keep the fastest time at which we can achieve each state we generate.

For each state, starting from the initial state where we have caught nothing, we try running in either direction until we catch the fastest uncaught quail in that direction, and then run back to the origin. We also try running to catch each quail farther than the fastest uncaught quail in each direction, and then running back. For each of these options, we compute the new fastest uncaught quail in the direction we ran, which will tell us what the new state is when we return to the origin. If we have achieved the new state with a faster time than we had for it before, we update the time for that state.

Whenever we generate the state where every quail has been caught, we check the time at which we caught the last quail. The fastest of these times is the answer.