Code Jam 2019 - Round 1A

Analysis: Golf Gophers

Test set 1

The difficulty of this problem stems from the fact that the windmills can loop around. If we have a windmill with 5 blades, and we find it with blade number 2 pointing downward in the morning, does that mean that 2 gophers rotated it? or that 7 did? or 12? In general, if a windmill has B blades, and we find it in position P, all that we can say for sure is that some number (P + K × B) of gophers (for some integer K) tampered with it during the night.

Because of this, we might reason that we should put as many blades as possible (that is, 18) on each windmill, and then hope that none of those windmills is visited by more than 17 gophers, in which case the total number of rotations of all blades equals the total number of gophers. This turns out to be a reasonable hope! It is hard to directly calculate the probability that none of the 18 blades will be turned more than 17 times on a given night, but a quick simulation can tell us that even in the worst case of 100 gophers, the probability of this happening is about 0.00017. Since we can run this same experiment 365 times and take the maximum result that we find, the chances of a wrong answer (due to getting a misleading result all 365 times) are infinitesimally small.

Test set 2

In test set 2, we only have 7 nights to work with, and the number of gophers can be quite large, so we cannot reasonably hope that the windmills will not loop. Now what? We might consider using different numbers of blades on each windmill during a given night, but it's hard to see how that buys us anything.

Suppose that, on one night, we try putting two blades on every windmill. At first this doesn't seem to help — shouldn't the resulting data should be almost pure noise? However, we can observe that if the number of gophers is odd, the total number of turns (across all windmills) will be odd, and if the number of gophers is even, that total number will be even. So we have a way of determining the parity of the number of gophers, no matter how they happen to turn the blades!

We can extend this idea to find out how many gophers there are modulo any number of our choice between 2 and 18. Since we only get 7 nights, though, we should choose our numbers carefully. One promising idea is to make them all prime, and there are seven primes in the range we can use: 2, 3, 5, 7, 11, 13, and 17. Then we can try to use the construction suggested by the Chinese remainder theorem to uniquely determine the number of gophers. However, this method would only work for any number of gophers up to 2 × 3 × ... × 17 = 510510; it cannot distinguish 510511 gophers from 1 gopher! We are in trouble, since we might have up to 10^6 gophers.

The final insight that we need is that the Chinese remainder theorem only requires its moduli to all be *relatively* prime, not necessarily prime. So we can use 16 in place of 2, and 9 in place of 3. This lets us identify any number of gophers up to 5 × 7 × 9 × 11 × 13 × 16 × 17 = 12252240, which easily covers what we need. (We can also get away with using the numbers 12 through 18, for example; we leave the details to the reader.)

Notice that we do not really need to do any calculations based on the Chinese remainder theorem; since the number of possible answers is relatively small, we can check all of them until we find the one that has the appropriate residue for each of our chosen moduli.

Test set 1 appendix

To prove that the probability of at least one blade rotating more than 17 times is about 0.00017, we can solve a related problem:

"Count the number of integer arrays of length **L** such that each array element is one of 1, 2, ..., **U** and each number appears at most **B** times in the array."

This related problem is equivalent to counting integer partitions with constraints on the size of each partition.

We can solve this related problem using Dynamic Programming. First, we choose how many times **U** appears in the array; say it appears K times, with 0 ≤ K ≤ min(**B**,**L**). Once we know K, we have to choose where those K values go. Using combinations, we can see that there are `C(`**`L`**`, K)` possible options. For each option, we can treat the **L**-K other spaces as an array that must contain the numbers 1, 2, ..., **U**-1 such that no number appears more than **B** times, which is a subproblem of our original problem, so we may recurse. If we sum over all possible values for K, we have the total number of valid arrays.

If we solve the above problem with **L** = 100, **U** = 18, **B** = 17, we see that there are 336647783248234011860927063629187654598455062446560501834487820535956663161762533555609870639313859125191714928476256 (or approximately 3.366×10^{125}) arrays such that no number appears more than 17 times. These arrays can represent which windmill the gophers select in a *good* configuration out of the 18^{100} possible configurations. This gives us the probability quoted above.