Code Jam 2013 - Round 1B

Analysis: Falling Diamonds

This was a tricky problem, and solving the small often made the difference between advancing and not - in particular, solving all of this problem was enough to advance; as was solving the small of this problem and the large of Osmos.

The small case

For the small case, we had only twenty diamonds to deal with in each test case. Note that every diamond will do something non-deterministic at most once, when it falls point down on the point of another diamond (when it can slide in one of two directions). This means that we will have at most 2^{20} different things that can happen when the diamonds fall, so we can simply try to enumerate all of them (note that there is actually less possible paths - for instance the first, fifth and sixth will never have a choice).

This is actually a bit tricky to do, since in each branch we have to keep track of what is the probability of reaching this branch — it is not only dependent on how many diamonds we processed so far, but rather on how many diamonds had a choice so far. After this is done, we need to figure out in how many of all the options the place we're interested in did get a diamond and add them all up. All this is not easy to get right, but it is doable, as over 900 of our contestants proved!

The large case

For the large test case, we will need to be smarter — simulating all the options is obviously not a choice for 10^6 diamonds. We will begin with the following observation:

> The diamonds fall in layers. First a diamond falls at (0, 0), then diamonds fall into positions with |X| + |Y| = 2, only after all five of these are filled the positions with |X| + |Y| = 4 start filling, only after them the |X| + |Y| = 6 start filling, and so on.

Indeed, note that the diamond sliding to one side does not change the layer it is in, since it always starts sliding in some (0, 2k) position, and the (0, 2k) position is always the last in a layer to be filled.

Thus, when **N** diamonds fall, the only uncertainty as to how they shape up is in the last layer, and this is what we have to calculate. If the place we are considering is not in this layer, we can respond immediately. Thus, we have only to figure out probabilities in the last layer.

The dynamic programming approach

First let's estimate how large the last layer can be. If we have at most a million diamonds, one can calculate there will be no more than 710 layers. When diamonds fall, the state of the layer can be described by two numbers — how many diamonds are on the left of the center (with negative X), and how many are to the right (we assume here there aren't enough diamonds to fill this layer, so the top spot with X = 0 will stay empty). This means that when the diamonds drop, there are roughly 500,000 different states to consider.

One can approach to this problem is dynamic programming. For each of the states possible for the last layer, we calculate the probability of reaching this state when the appropriate number of

diamonds has dropped (each state determines the number of diamonds uniquely).

A formulaic approach

One can also notice that what matters is how many diamonds of the ones that hit the top decide to go left, and how many to go right. Which diamonds exactly are those does not matter for the final state. Thus, we can precalculate binomial coefficients (or rather binomial coefficients divided by 2^D, where D is the number of diamonds falling into the layer), and — once we know which layer we're looking — sum up the options that lead to a diamond falling into the right place.