Code Jam 2015 - Round 2

Analysis: Drum Decorator

Some important initial insights are:

- A cell can only take three values: 1, 2, or 3. 4s would have to keep propagating until they hit an edge cell, and edge cells only border three other cells.
- If a 3 appears, it must be in a band of 2 complete rows. To see this, place a 3 along the border of the top of the drum, and note that the 3 adjacent cells must all contain 3s. This must continue all the way around the top, which also fills up the row below. Suppose that some other finite arrangement of 3s was valid — then there must be a highest 3 in the pattern, above which no other 3 can be placed, but then that 3 can only be part of a set of 2 rows of 3s, by the same argument above.
- If a 2 appears, it must be either in a 2x2 square of 2s, or in a (possibly winding) line of 2s that goes all the way around the drum and reconnects with itself.
- 1s can only appear in adjacent "dominoes" of two.

Besides the bands of 3s, there are four patterns using 1s and 2s:

I. one-thick bands of 2s:

2...

II. alternating upright dominoes of 1s and 2x2 squares of 2s:

122...
122...

III. a line of 2s winding around horizontal dominoes of 1s:

222112...
112222...

IV. a line of 2s winding around vertical dominoes of 1s:

2212...
1212...
1222...

None of these patterns can border each other, but they can be separated by bands of 3s. The number of columns in a drum can rule out some patterns -- II, III, and IV require multiples of 3, 6, and 4, respectively.

Now we can use dynamic programming to assign patterns to the drum. Our state is how many rows we have filled, and whether the previous pattern was a pair of rows of 3s.

Each of the five patterns repeat with some period P, which is either 1, 3, 4 or 6, so there are P ways to place that pattern on the drum.

Each overall drum assignment also has a period, which is the least common multiplier of the periods of all of the patterns on it.

An extra complication is that drum assignments that are rotations of each other should only be counted once. So our dynamic programming state should also include the period of the current

drum assignment. When counting the final number of solutions, we divide the number of solutions in each state by the state's period. For example, if we have 18 ways to produce a solution of period 3, this only corresponds to 6 solutions.

Another way to handle equivalent patterns is to use Burnside's lemma. We leave the details as an exercise to the reader!