Code Jam 2020 - Round 1A

Analysis: Square Dance

Test Set 1

Test Set 1 can be solved by simulating the competition:

1. Iterate through all the cells of the floor. For each cell, if there is a dancer in that cell:
    - Add their level to the interest level of the competition.
    - Loop through their row and column to find all of their compass neighbors.
    - Count the number of their compass neighbors, and sum up their levels.
    - If the average level of the neighbors (calculated as sum of their levels divided by how many neighbors there are) is greater than the level of the current dancer, add this dancer to an elimination list.
2. Go through the elimination list and eliminate the dancers.
3. If the elimination list is empty, we are done. Otherwise, start again from step 1.

Each iteration of the above algorithm has a time complexity of O(**R**×**C**×(**R** + **C**)), and there are as many iterations as there are rounds in the competition. The number of rounds in a competition cannot be larger than the total number of dancers, **R** × **C**. So the total time complexity is O($\mathbf{R}^2$×$\mathbf{C}^2$×(**R** + **C**)).

Test Set 2

Let's optimize our Test Set 1 solution — specifically, the following parts of it:

- For each round, we iterate through all **R**×**C** cells and check whether each dancer is eliminated.
- For each dancer, we may have to iterate through a significant part of their row and column in order to find their compass neighbors.

To improve the situation described in the first point above, we can observe that for i > 1, if dancer d is not eliminated in round i-1 and they are eliminated in round i, their set of compass neighbors must have changed in between. For that to have happened, at least one compass neighbor of d must have been eliminated in round i-1.

Using this observation, instead of checking every dancer in round i, we can limit the check to the compass neighbors of those eliminated in round i-1. Since a dancer has at most 4 compass neighbors at the time of elimination, the length of the list of candidates to check in round i is at most 4 times the length of the elimination list for round i-1. The sum of the lengths of all elimination lists is at most **R** × **C**; therefore, the sum of the lengths of all lists of candidates to check is at most 4 × **R** × **C**. With the initial check of all dancers, this means that we do O(**R** × **C**) checks overall with this improvement.

To optimize finding compass neighbors, notice that when we remove a dancer, their eastern neighbor becomes the eastern neighbor of their western neighbor and vice versa. The same thing happens to the southern and northern neighbors. So, instead of finding the neighbors each time we need them, we can maintain the specific location of the neighbor in each direction for each dancer, and update their locations in constant time whenever we eliminate one. This is effectively like keeping a linked list to represent each row and column, and this structure allows us to remove and find neighbors in constant time.

Since we do O(**R** × **C**) checks thanks to the first optimization, and each check can be done in constant time thanks to the second optimization, this yields an O(**R** × **C**) time algorithm that solves the problem fast enough for Test Set 2.