1. Preliminaries and rules

Today's task is given to you in a short and precise format. By purpose, the scope of the task is very large and it is almost impossible to implement a complete solution within the given time frame. Please invest not more than 8 hours to complete this task.

Please proceed in an incremental way using baby steps (MVP like). Ensure that at any point in time, your solution can be executed and add small features one by one. **In this way, the time limit will not hinder you to deliver at least a partial solution – which is very acceptable!**

1. Definition of Done

The definition of done for any feature is:

- You can reuse code parts from the Internet (GitHub, Stack overflow, your own, etc :)) BUT you should mention this in the README or in the interview
- For the front-end role focus should be (can be) on the UI. Game algorithm can be mocked or partially mocked.
- For the back-end role you can skip UI and cover game cases via Unit Test
- Unit tests are given that demonstrate the correctness of your code (100% code coverage is not required)
- The code adheres to high quality standards (as few lines of code as possible), best practices and a clear structure. The code is easy to read without help of comments and uses object oriented features such as inheritance and design patterns to make the code extensible and easy to maintain
- The resulting software is functional and can be tested by us
- If don't have time to implement something/ or you have an ideas how to improve it you can mentioned in Readme, as comments in the code, in the interview
- Solution should be pushed to the GitHub, GitLab, Bitbucket etc.

1. Requirements

You are requested to implement a popular one player game (2024 https://play2048.co/), which is available on many smart phones.

Game algorithm

https://en.wikipedia.org/wiki/2048_(video_game)

The player is given an n x n board of tiles where each tile is given one of m colors. Each tile is connected to up to four adjacent tiles in the North, South, East, and West directions. A tile is connected to the origin (the tile in the upper left corner) if it has the same color as the origin and there is a path to the origin consisting only of tiles of this color. A player does a move by choosing one of the m colors. After the choice is made, all tiles that are connected to the origin are changed to the chosen color. The game proceeds until all tiles of the board have the same color. The goal of the game is to change all the tiles to the same color, preferably with the fewest number of moves possible. It has been proven that finding the optimal moves is a very hard computational problem. It has also been shown that finding the minimum number of flooding operations is NPhard for m > 3. This even holds true when the player can perform flooding operationsfrom any position on the board. However, this variant can be solved in polynomial timefor the special case of m = 2. For an unbounded number of colors, even this variant remains NP-hard for boards of a dimension of at least n = 3 and is solvable in polynomial time for boards of dimension n = 2