2048 probability model

TODO: 4 tile spawn rate varies between versions, needs to be encoded as a variable.

Movement algorithm:

left/right: For each row, start in the 2nd last column. Take the tile, and move it in the direction of the swipe until it hits another tile, if they have same value, then fuse them. Repeat for all rows.

Tiles cannot be fused consecutively

so 2 2 4 ← turns to 4, 4 not 8

but 2, 2, 2, 2 ← 4, 4 because the 3rd to will hit the 4, so it is not consecutive.

Q: How much data do we need to store for each tile? Max tile value is 2^{N^2+1} so a 2x2 would be 2^5, 3x3 : 2^{10}, 4x4: 2^{17}

Number of bits required is $\log_2(N^2+2)$

$2^0, 2^1, \ldots 2^{N+2}$

↖ represent blank tile

Given the problem space, 2x2 and 3x3 game modes should be solvable. The table base will be a graph similar to a Markov Matrix

Suppose $a = 0.2$ is the chance of solving. Lets look at the first 3 layers of the graph for 2x2

$$\begin{tikzcd}
\text{level 0} & & & 0000 \arrow[dlll, "0.2"'] \arrow[dll, "0.05"] \arrow[d, "0.2"] \arrow[dr, "0.05"] \arrow[drr, "0.2"] \arrow[drrr, "0.05"] \arrow[drrrr, "0.2"] \arrow[drrrrr, "0.05"] & & & & & \\
\text{level 1} & 0002 \arrow[dd, "L"'] \arrow[ddr, "U"] & 0004 & 0020 & 0040 & 0200 & 0400 & 2000 & 4000 \\
 & & & \cdots & & & & & \\
\text{level 2} & 0020 & 0200 & & & & & &
\end{tikzcd}$$

Nodes in odd levels represent the intermediate grid state between swiping and the new tile spawning. Each edge between an even and odd node has a weight of $\frac{(1-a)}{e}$ or $\frac{a}{e}$, for spawned 2 and 4 respectively, where e is the number of empty slot where a tile can spawn. Note from the example that the nodes in level 2 are actually duplicates of nodes in level 1 so it is technically incorrect to label nodes with levels, the edges pertain to levels, not the nodes, but it still can be convenient to think that way.

Odd edges will be weighted with a sentinel value (-1.0) to represent a non-random transition.

Data Structure:

I'll use a hashmap for nodes and edges

Node: $\overbrace{\text{grid}}^{\text{key}}, \overbrace{\text{score}}^{\text{value}}$

Edge: $\overbrace{\text{grid1, grid2}}^{\text{key}}, \overbrace{\text{weight}}^{\text{value}}$

Obvisously, these tables, so for when we can't fit it all in memory, we'll use SQLite

The tablebase interface will look something like this:

```
ITablebase<N> {
    init(GridState initState, int maxDeph)
    queryScore(GridState s)
}
```