Michael Perez

Cella Rule 45 Complexity Order
Team MPP

The bottleneck of this application is in draw-stuff.js, in the function draw_cells(rctx).

There are three double for-loops in this function:
- One to initialize all of the arrays and create the cell objects
- One to apply Rule 45 to all of the cells
- One to draw all of the cells

The loop sizes are determined by the input height (n) and width (m) of the grid.

In the first double for-loop, the following significant operations occur:
- Initialize n+m arrays: O(n)
- Create cell objects at every index of the 2D array: $O(n^2)$

By nature of dominating terms, this first double for-loop is $O(n^2)$.

In the second double for-loop, there are only O(1) operations, so the second double for-loop is $O(n^2)$.

In the third double for-loop, the only significant operation is fillRect. In my research, I was unable to find the time complexity of it, but depending on what it is, here are the following time complexities of the third double for-loop:
- If O(1), $T(n) = n(n(1)) = n^2 = O(n^2)$
- If O(n), $T(n) = n(n(n)) = n^3 = O(n^3)$
- If $O(n^2)$, $T(n) = n(n(n^2) = n^4 = O(n^4)$

The overall time complexity of this application is $O(n^2)$, or higher depending on what the time complexity of fillRect is.