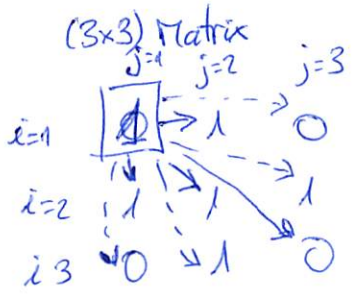
First Attempt



① You need to iterate around each sorrounding element of the current position (0,0). If all sorrounded values are '1's we have a (2x2) array with (4) dimension

② Now, you have to iterate around "second level" of the sorrounded elements. (- - ->). In this case, there is no extra-dimension.

③ Then, we make the same iteration as before with element (0,1), (1,0), (1,1) and so on.

For example

Again, iterate around the nine sorrounded elements. In this case, maximum dimension is $3 \times 1 = 3$.

Drawbacks: → Extremely difficult to implement
→ loads of iterations
→ we need temporally states to save/load results.

Working Solution

→ [1x6] Matrix. For example, [0 [1,1,1] 0,1] → Max dimensió 3x1

→ Now with a [6x1] Matrix: $\begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}$ → Max dimensió 1x4

→ For the time being, we can use the same solution for multidimen. arrays (mxn)

Example ① [0,1,1] → Max Dimen = 1x2 → temp_array << 2

② $\begin{bmatrix} 0 & 1 & 1 \\ 1 & 2 & 2 \end{bmatrix}$ → Row i=2; if we find '1' we ~~them~~ add the value of the element (i, j-1)
→ Max Dimen = (2x2) → temp_array << 4

i=1 / i=2 / i=3 $\begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

③ $\begin{bmatrix} 0 & 1 & 1 \\ 1 & 2 & 2 \\ 2 & 3 & 3 \end{bmatrix}$ → Row i=3; Max Dimen = (3x2) → temp_array ~~<<~~ 6

(4) So, we have this temp_array = [2, 4, 6]

This is the maximum dimension of a sub-array inside the array !!

(5) In order to not change the origin array, we need to create a copy of it, so we can change values in order to save them in the correct position.

(6) I have comment ruby-file called "m.rb" for additional information.