An "O" can be replaced by "X" if there is an "O" in its neighborhood
but that neighborhood "O" should also be a valid "O" means, it should neither be on boundary
not it is connected to some contiguous "O" that ends at boundary.

Approach 1:

we could have done this using two nested for loops, iterating over each element that
does not lie on boundaries and its neighboring elements to be "X" or "O".
if neighboring elements are all "X". we could have simple replace that "O". but if any
of the neighboring element is "O", then we have to make a dfs call over that to detect
the chain of "O"s ending at the boundary. We will take a Boolean helper matrix of the same size to
keep track of the information about a particular path if it ends at the boundary or not.

in future if any of the cell contains "O", it can directly check for its validity.

time complexity: O (M x N)
space complexity: O (M x N)

Approach 2:
instead of approaching from outside of the matrix to inside to detect the boundary,
we can traverse the boundaries of the matrix for "O" and use dfs to find the contiguous
"O"s inside the matrix and replace those by any special character so that we couldn't
trace the same sequence in the next turn.

Finally, we traverse the matrix and replace all "O"s with "X"s, because they are the valid ones.

and all special characters with "O"s.

time complexity: O (M x N)

Auxiliary space complexity (using stack): O (M x N)