* Problem Statement :

Each cell in a matrix of size n\*n contains either forest or a desert. Each year a desert converts all its neighbors (8 connected) into desert. How many years will it take to run out of forests completely?

* This problem can be compared to the Computer Graphics flood fill algorithm. In such a domain the required result is actually the amount of iterations flood fill algorithm is going to take to completely paint the screen.

To solve the problem it is necessary to design an algorithm to efficiently fill in deserts in minimum number of years.

Algorithm:

1. START.
2. Count number of deserts present and put’em into a pool (call these ‘seeds’).
3. If no deserts are present or number of deserts equals n\*n, STOP.
4. Check for number of forest neighbors for each element in the pool (array).
5. Select the one with highest neighbors and apply 8-connected conversion.
6. Update year value by 1.
7. Go to step 2.

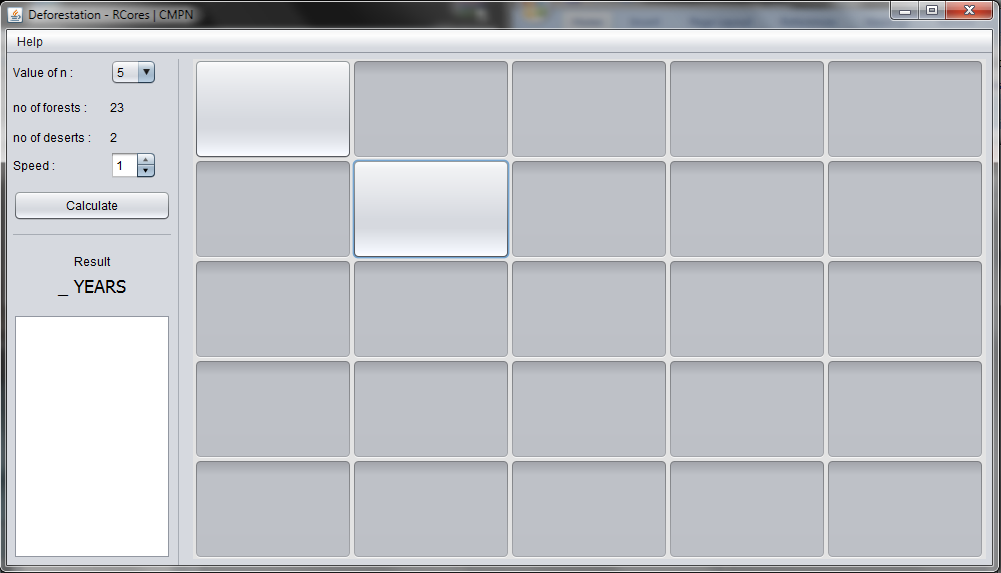
Graphical Representation:

The following java program simulates the above problem and finds the minimum number of years. The application need Java Runtime Environment 1.7 or above.



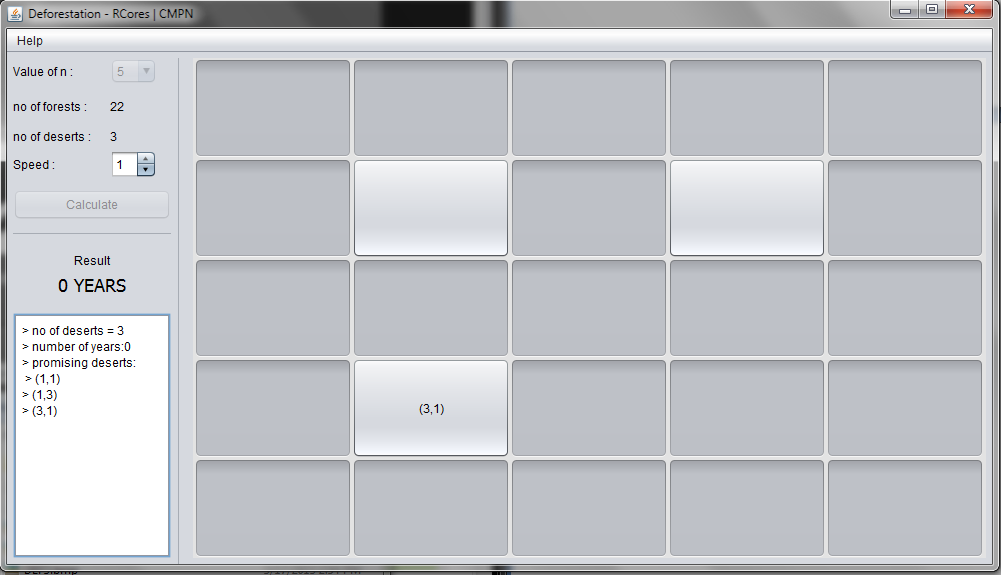


Select the order of matrix from the selector “Value of n”

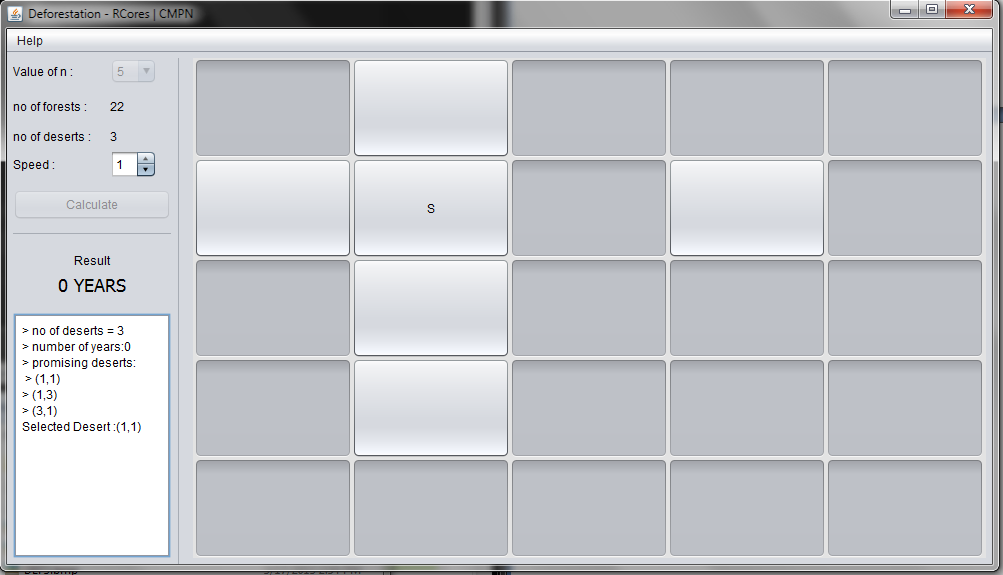


Select the cells to make them forests.

The gray areas are deserts and bluish areas are forests.

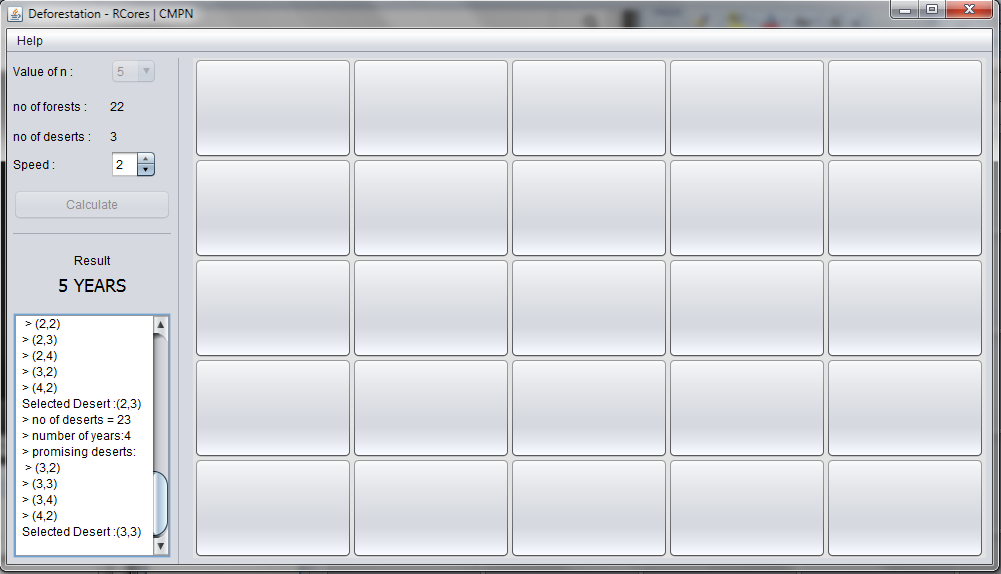


Hit the Calculate button to start simulation.

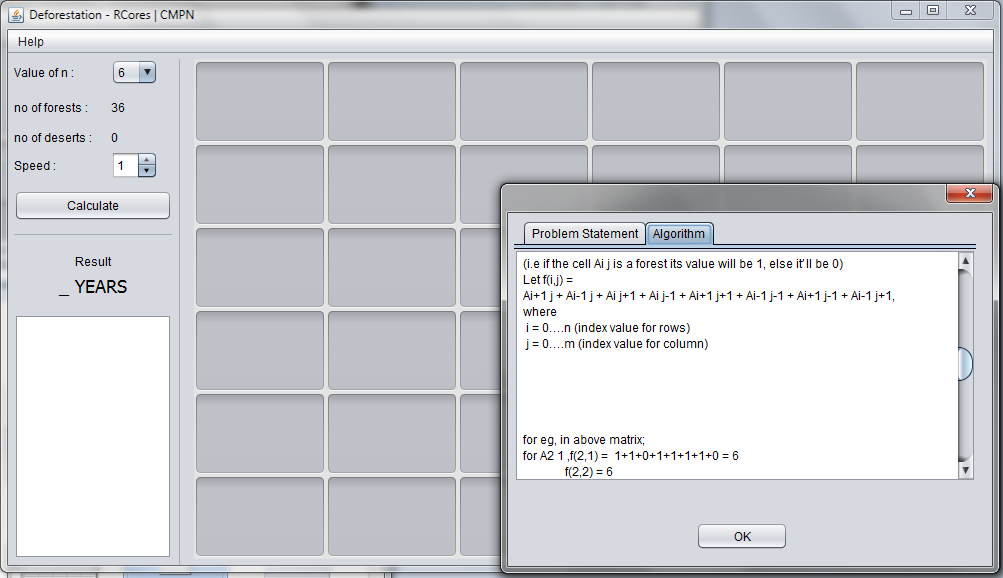


Vary the speed using the spinner “Speed”.

Do not interfere with the matrix buttons while calculation is in process.



Once all the cells are filled with deserts, the result will be displayed below the “result” label.



Find the algorithm in description by going to Help>About>Problem Statement on the menu bar.