**Forest Fire Simulation Report**

Contents

[**Introduction** 1](#_Toc90080668)

[**Base Model Design** 1](#_Toc90080669)

[**Array and Time-Step Design** 1](#_Toc90080670)

[**Random Chances** 1](#_Toc90080671)

[**Cell Interaction** 1](#_Toc90080672)

# **Introduction**

Simulations run through python can, depending on the parameters chosen, accurately simulate what a real-world scenario may look like. Therefore, it becomes possible tom use simulations to mimic how natural disasters such as forest fires may spread. The simulations run within this project therefore aim to mimic how a fire may spread through a forest, with random effects being considered and accounted for.

# **Base Model Design**

## **Array and Time-Step Design**

This simulation will design the array in one of two ways, either by the default values or the user-given values. The default values of N=50, and 1000 time steps will lead to the creation of a 50 by 50 array, with 1000 steps then being run per simulation. N being 50, or the user inputted value, will create the initial array with 50 columns and 50 rows; this array is then filled with 0s and 1s randomly. Random generation of the initial array was chosen to remove bias, and make the simulations run more accurate of how the real-world scenario would be.

## **Random Chances**

For the base simulation, two random parameters were used. In the second simulation, a further random effect was added; this was the chance of rain, which will be explained further in the report. Of the two random parameters used within the base simulation, the first used was the chance of lightening randomly occurring. This will turn a 'tree' cell to a 'fire' cell, thus creating a mechanism through which the simulations are unable to become filled with only 'tree' cells. The next was the random chance of a tree growing in an empty cell, thus turning an 'empty' cell to a tree cell. Through this random effect, the simulations are therefore unable to become filled with only 'empty' cells. These random chances are how the initial time-step array becomes distinctly different at each step iteration.

## **Cell Interaction**

For the simulations to run, logic must be created behind how cells are able to 'see' each other, and therefore how they can interpret one another. This process however needs to be able to compensate for when the cell is located within the top or bottom row, or when the cell is at the very right- or left-hand side.

These simulations run based on the following cell interaction logic:

* If a cell is within the top row, then the cell above it is considered equal to the state of the cell itself. For example, if the cell being analysed was a 0 (empty), then the cell above would be considered a 0 as well, thus leading to no direct effect on the original cell.
* If a cell is within the bottom row, then the cell below it is considered equal to the state of the cell itself. For example, if the cell being analysed was a 0 (empty), then the cell below would be considered a 0 as well, thus leading to no direct effect on the original cell.
* If the cell is located at the right most-hand side, then the cell to the right is considered to be in the same state as the original cell itself. For example, if the cell being analysed was a 0 (empty), then the cell to the right would be considered a 0 as well, thus leading to no direct effect on the original cell.
* If the cell is located at the left most-hand side, then the cell to the left is considered to be in the same state as the original cell itself. For example, if the cell being analysed was a 0 (empty), then the cell to the left would be considered a 0 as well, thus leading to no direct effect on the original cell.
* Corner cells are treated as the previous logic explains.