Report on the Simulation of Forest Fire Spread

# Overview

This report aims to describe the processes taken to simulate the spread of fire on an initial forest site filled with burning and non-burning trees. The spread of fire at every location of the forest could be influenced by a number of factors which might include, the closeness of one of the trees in the forest to a burning tree in its Moore neighbourhoods or the possibility of lightning strikes on the forest site. In addition to that, some types of trees are immune to fire such that even though there may be burning influences, they don’t get burnt easily. As a result, in the following sections, I aim to describe how this simulation has been achieved according to the coursework specification.

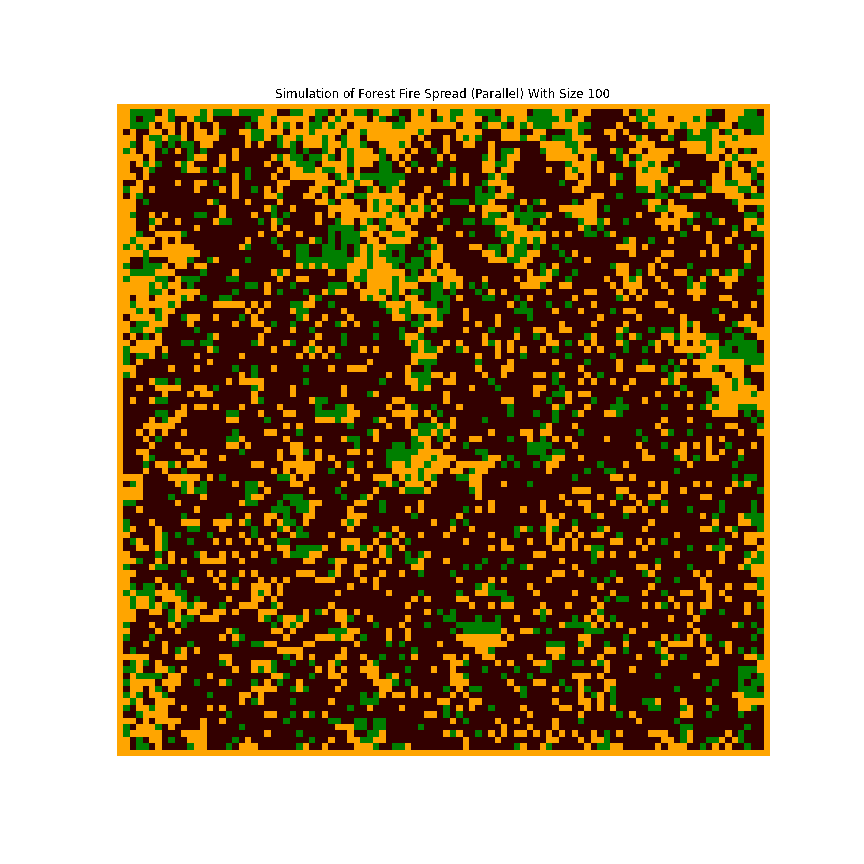


Figure 1- a forest site burning

# Simulation Process

To commence the simulation, I needed to gather some details needed to initialize the forest site. This was important before spreading the fire on the site. As a result, this was done with the Numpy library in python. I also created variables for the various probabilities given in the coursework and set values to be inserted in the Numpy grid to represent burning and non-burning trees, empty land, and border. Also based on the assumption that trees are green, and fire red, I picked close colours to represent them in the simulation. Finally, I also assumed the fire flames burn in 300 milliseconds, represented by the animation delay. The image of the section is attached below.

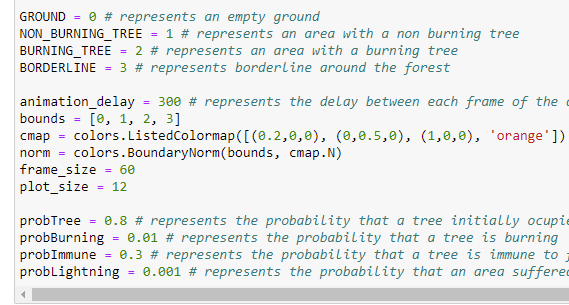


Figure 2- Gather details for the simulation

To Initialize the forest I created an empty forest site grid depending on the size of the forest given, and filled it with empty lands, and burning and non-burning trees using two probabilities, probTree and probBurning as specified in the specification.

As a result, if at any given location in the forest grid, the probability that there is a tree there is true, the probability that that tree is burning is checked. If the tree is not burning, then, we just set the tree value, otherwise, we set the location as having a burning tree.

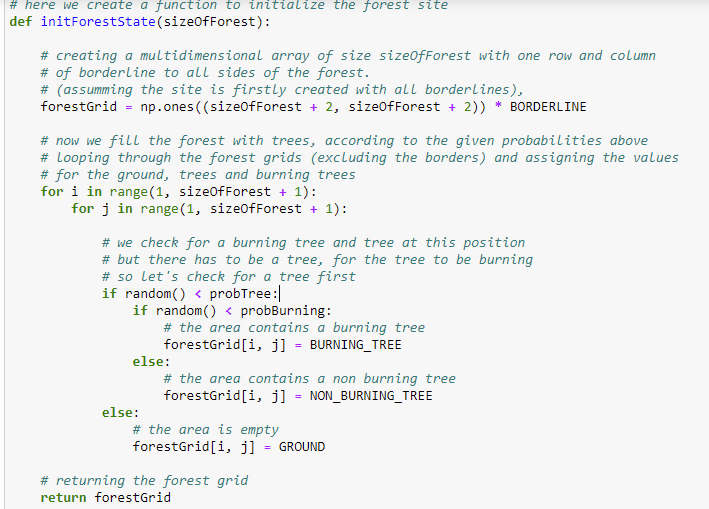


Figure 3- initializing the forest site

Thereafter, we apply the spread rules to the site. But before then, to avoid boundary effects that might prevent us from successfully applying the spread at the borders, I extended the boundaries of the site using the periodic boundary conditions (Harris & Stocker, 1997). This is done by creating ghost areas of opposite sides to make it look like they are both boundaries are connected. Once the ghost areas have been created at the top, left, right and bottom, we then move to apply the spread on the site.

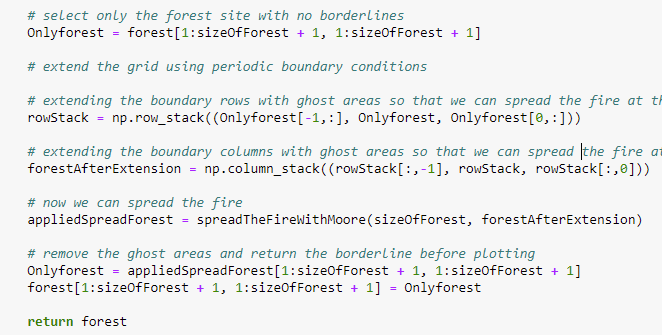
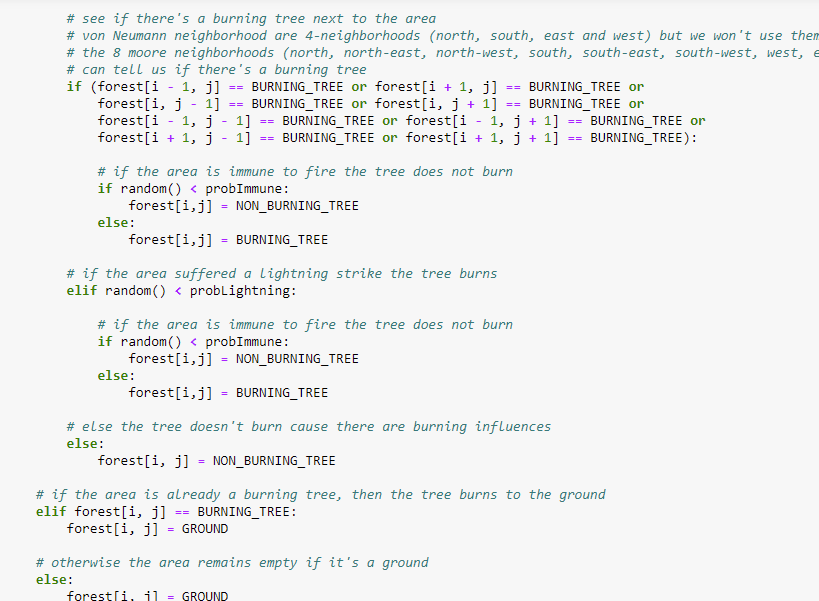


Figure 4 - extending the site boundaries with ghost cells

The spread of fire at a given area on the forest site is determined by burning influences around and the nature of trees in the area. Hence, according to specification, to spread the fire, during a loop across the site, if an area has trees, we check if any of its Moore neighbours (that is, trees at its north, north-east, east, south-east, south, south-west, west, and north-west) (Shiflet & Shiflet, 2006) contains a burning tree. If one of them contains a burning tree and the tree is not immune, it burns, otherwise, it does not burn. In the same vein, if the probability of lightning strikes is true, and the tree is not immune, it burns, otherwise it does not burn.

And else if an area has a burning tree, we set the grid position as an empty land, because the tree burns to the ground, otherwise, the remaining area on the site remains empty if there’s nothing there initially. Please see the figure below.



To visualize each iteration of the spread of fire, I used the animate.FunctionAnimation function in Matplotlib to animate the forest fires. It first displays the initialized site and then calls the VisualizeForestFireSpread function which expands forest boundaries and spreads the fire at every iteration. The spread was applied 60 times for each of the forest sizes. Please see the figure below for the code.

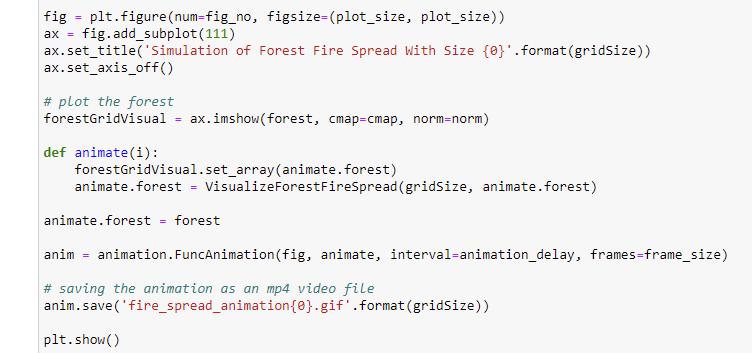


Figure 5 – animation

# Implementations, Results and Evaluations

The Forest fire simulation was implemented in two ways, sequentially and by means of parallelization. The essence of parallelization is the help utilize the processing resources available on the computer efficiently and make the simulations fast. By parallelizing the functions, significant improvements were recorded. See the table below for the time used for a single iteration of forest for each forest size. That is, the time in seconds it takes to initialize the forest, extend the boundaries of the site and spread the fire. This was captured for all forest sizes as displayed in the table below.

|  |  |  |
| --- | --- | --- |
| Forest Sizes | Sequential | Parallel |
| 100 \* 100 | 0.085 | 0.061 |
| 400 \* 400 | 0.65 | 0.61 |
| 800 \* 800 | 2.66 | 2.44 |
| 1000 \* 1000 | 4.66 | 3.76 |
| 1200 \* 1200 | 6.52 | 5.92 |
| 2000 \* 2000 | 17.28 | 17.07 |

The parallelization was implemented using the Numba python package that works by compiling the functions before executing. And time was captured using the time python library before and after the execution. Below is an example of how a function was parallelized with Numba.

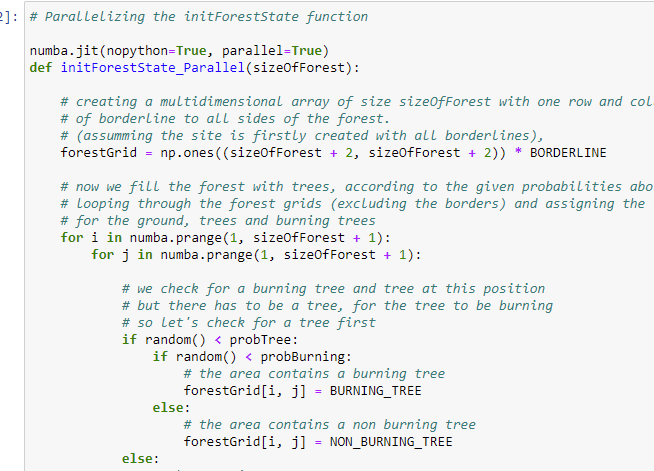


Figure 6 – Parallelization

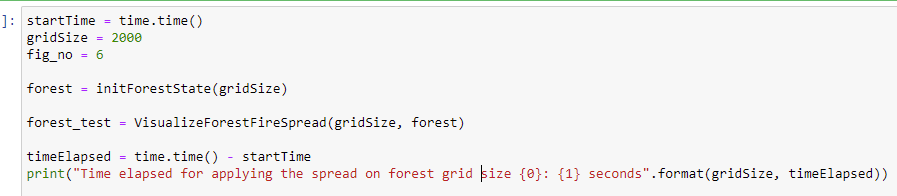


Figure 7 – How time was captured

# Conclusion

The simulation exercise shows that parallelization improves simulation time, which is a new exposure for me from this activity and seeing how much time difference it makes. However, for simulation of larger forest sites, I would recommend using faster systems with more processing cores to make the speed faster.

# References

Harris, J. & Stocker, H. (1997) *The handbook of mathematics and computational science*. New York: Springer.

Shiflet, A. B. & Shiflet, G. W. (2006) *Introduction to computational science: Modeling and simulation for the sciences*. Princeton; Oxford: Princeton University Press.