# SCIF30005: Forest Fire Mini Project

## Overview

The topic of the project is the forest fire model, a type of cellular automaton which is introduced in the 'Forest Fire' video, found on Blackboard under week 19. The rules are briefly explained below, but you should watch the video for a more detailed explanation.

The deadline for submission is 12:30pm, on Wednesday of week 23. You should submit:

- your C++ code
- a 5-6 page report

You will not be penalised for going slightly over the page limit, but if you are significantly over, you should consider whether you are e.g. including more graphs than is necessary to justify your conclusions. You do *not* need to submit any code used to generate graphs or otherwise post-process your output, or any bash scripts that you may have used.

## Model Rules

A square grid of size $N$ is randomly filled with trees, with probability $p$ that any given cell contains a tree. To start the fire, any living trees in the top row are then set on fire. Then, for each point on the grid at time step $t$:

- any living tree next to a tree which was burning at step $t-1$ starts burning
- any tree which was burning at step $t-1$ is now dead
- all other sites stay in the same state (i.e. empty sites remain empty, living trees with no burning neighbours remain alive, dead trees remain dead).

A neighbour is defined as a cell to the left, right, above or below the cell we are interested in, with diagonal cells *not* considered to be neighbours (i.e. we are using a Von Neumann neighbourhood). The simulation proceeds until no more trees are burning.

## Implementation

You should implement the forest fire model in C++, using MPI to parallelise a *single run* of the model, i.e. it should be possible to run the code for a single starting grid, grid size $N$, and probability $p$, while using multiple MPI tasks.

The C++ code should contain the following features:

- the ability to read in an initial grid from a text file, for which an example will be supplied
- the ability to generate a random starting grid of size $N$ with probability $p$
- the ability to average over $M$ independent runs, e.g. for $M$ starting grids which were initialised with different random seeds
- the code should output:
  - the number of steps before the fire stops burning
  - whether or not the fire reached the bottom of the grid
  - the time taken to run a given simulation

You will also need to run the code for different probabilities $p$ and grid sizes $N$. It is up to you whether you do this within your C++ code (e.g. using a loop over probabilities $p$), or by using a bash script to run your code repeatedly with different inputs.

## Model Convergence

In the video introducing the forest fire model, we talked about convergence with respect to a) the number of random starting grids, $M$ (i.e. the number of repeat runs) and b) the size of the grid, $N$. For example, if you run the code once with a given random starting grid, and then run it again with a different starting grid, the number of steps the simulation runs for may not be the same, even for the same $N$ and $p$ values, due to the effects of randomness. If, however, you run the code $M$ times and average the results, for a large enough value of $M$ the average number of steps taken will no longer change significantly. I.e. for a large enough value of $M$, further increases in $M$ will result in only negligible changes to the average behaviour. In other words, the model has converged.

For this project, we will focus only on convergence with respect to the number of repeat runs, $M$. You may use a fixed grid size of $N = 100$. Using your code, you should answer the following questions:

- is $M = 50$ a sufficient number of repeats to reach convergence?
- does the answer depend on the initial probability $p$, and if so, how?

Your answer should be backed up by data, and you should describe how you came to this conclusion, as well as any strengths and/or limitations of your approach. For example, did you have to make any compromises about the number of calculations performed, and given unlimited time, would you therefore do any further calculations?

## Performance Analysis

The final aspect of the project is to explore the parallel performance of the model using BlueCrystal4. For this part of the project, we are not interested in the behaviour of the model itself, only the time taken to complete a run, and how this varies with the number of MPI tasks.

Since the performance is intrinsically linked to the problem size, you should generate timing data for three different values of $N$:

- $N = 50$
- $N = 100$
- $N = 500$

You may run these calculations using $p = 0.6$, averaging over $M = 50$ runs.

Present and comment on your results, including a discussion of how the grid size influences the performance and why, as well as the metrics you have used to assess the performance (walltime, speedup etc.). State the maximum number of MPI tasks you would use for each of the three grid sizes, and why. Your discussion should also include any factors from your MPI implementation which impact on performance, e.g. your choice of MPI communication scheme and data distribution.

## Report

This is a mini project, and so the mark scheme takes into account both technical aspects, i.e. your code, including correctness, use of C++, choice of MPI implementation etc., and how you have used your code to answer the questions, i.e. your report. Make sure you read the instructions carefully and answer all questions in your report. You should also include sufficient data to backup your conclusions, as well as a discussion of any limitations, including potential future work that could be performed to address any of these limitations or extend the project.