# ABM specific part

1. The provided notebook contains a basic implementation of a forest fire model. The model consists of a grid. Each cell can be occupied by a single tree. Whether or not a cell is occupied by a tree is controlled by the density keyword argument. A tree can be either healthy, burning, or dead. If a tree is burning, the fire will spread to all neighboring healthy trees.

The provided model contains a complete implementation for the Tree class, but the model initialization is missing. Implement this.

self.scheduler = RandomActivation(self)

self.space = SingleGrid(height, width, torus=False)

for i, (\_, x, y) in enumerate(self.space.coord\_iter()):

if self.random.random() < density:

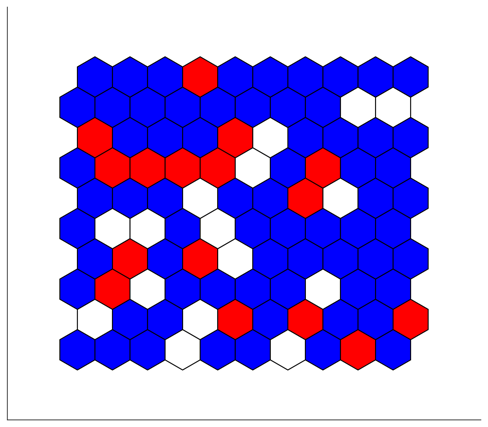
# Create a tree

tree = Tree(i, (x, y), self)

self.space.\_place\_agent((x, y), tree)

self.scheduler.add(tree)

1. Below, the image shows an alternative implementation for a forest fire model using hexagonal tiling. How would you approach implementing such a tiling? In your answer, indicate what, if anything, would need to be changed in the model itself, as well as what if any, additional classes, methods, or functions would be needed and how you would approach implementing those.



The best way to implement this would be to create a new HexGrid class by extending the Grid class from within MESA. In this way, only self.space in the model initialization would have to be changed. This new HexGrid class would need to abide by the same API as SingleGrid, but would use a different way of determining its neighbors. Moreover, in a hexgrid, there would not be a need to distinguish between von Neumann and Moore neighborhoods.

1. A key factor controlling fire spread is wind and wind direction. How would you expand the model conceptually to be able to account for this?

Currently, all neighboring grid cells will start burning. If you want to add wind direction, one way of doing so would be to make it random whether a neighboring grid cell would start burning. Next, the grid cells downwind from the current grid cell would have a higher chance of catching fire than grid cells that are upwind.

Implementation-wise, the main challenge of implementing something like this is that the current methods for getting the neighbors of a grid cell do not contain any information of whether a given neighbor is to the left, right, top, or bottom. So, it is likely that you would need to expand SingleGrid with a new method that would return information on the neighbors that does include this information.

1. A major structural uncertainty in the Forrest Fire model is what counts as the neighborhood of the tree. How does the behavior of the model change if you switch from a von Neumann neighborhood to a Moore neighborhood?
   1. List the modifications you need to make to the model.

Moore is used within the Tree class step method, so we must be able to change this easily. This can be done in various ways. A simple solution is to add a class Attribute for this to the Tree and set this class attribute from within the model.\_\_init\_\_.

* 1. How does the behavior of the model change depending on whether you are using von Neuman or Moore? Show the difference in behavior in an appropriate visualization and put in the code used to generate the data being visualized underneath it.

See notebook on github. Basically, you can run both versions of the model for the same seed and same number of steps. Visualize the end state after e.g., 40 steps. One step more sophisticated would be to count e.g., the healthy trees for each tick and visualize this as a plot over time. You will see that for a moore neighborhood, the number of healthy trees drops quicker.

* 1. And what happens to the behavior of the model if you change the radius (1, 2, or 3) in the case of a Moore neighborhood? Show the difference in behavior in an appropriate visualization and put in the code used to generate the data being visualized underneath it.

This is basically the same idea as the previous question. We can add a radius class attribute to Tree. However, we need to modify slightly the way in which we get our neighbors. Strangely, SingleGrid.neighbor\_iter does not take an argument to control the radius. However, by pulling up the code using ??, we can easily write the necessary code our self.

Next, we only need to run the model for 2 or 3 timesteps to see the profound differences that this makes. Again see Github for implementation details

* 1. Explain both results

The more neighbors, the faster the fire can spread, and thus the faster the entire grid is burned out. This will be robust across different seeds