Forest Fire Propagation

Elements

A forest is represented by an n*m grid, where each cell has one of the following possible types:

- WATER
- GRASS
- DENSE_TREES
- FOREST
- BURNING
- BURNED

- BURNING cells can only become BURNED
- WATER and BURNED cells remain the same
- the othe types of cells can become BURNING

Every type of cell has a predefined color, stored in the dictionray `cell_colors`

Each type of cell has a different probability of igniting based on the type of vegetation. These probabilities are stored in the dictionary `igniting_probabilities`. Each cell also has a very small probability of self ignite, stores in constant `self_ignite`

Based on the *number* of burning neighbors, the probability of a cell igniting is *multiplied* by a factor, stored in `burning_neighbours_factor`. It's an array where, at position 0, there is the multiplicative factor that will be multiplied if I have 0 burning neighbours, up until a maximum of 8 neighbours.

Other constants that are required for the additional features are the maximum possible elevation of the forest, stored in `max_elevation` and the maximum speed of the wind, stored in `max_wind_speed`. Also a dictionary, mapping from String to 2-tuple to make the wind-setting more intuitive :)

1.a `initialize_grid` function

Outputs a `m x n` grid, which represents the forest. Each cell has a specific i,j coordinate and a type. In particular, the grid will be a grid of `CellTypes`, enum defined above. At first, I store the possible types of cells in an array so i can use function random.choices. Because we want the initial grid to be randomly generated, we have a set of probability weights, these weights determine the likelihood of a cell to be of one of the possible types. In the beginning, no cell is burned already

`initialize_grid` pseudocode

`initialize_grid` code

1.b `plot_grid` function

Visualize the current state of the grid using different colors for each type of area using the PIL library. The pseudocode represents is the logic i would use, the actual plot_grid function is implemented using Pillow

`plot_grid` pseudocode

`plot_grid` code

1.c `neighbours` function

Given the coordinates of a cell and the grid, returns an array of coordinates of all its neighbours.

`neighbours` pseudocode

`neighbours` code

1.d `propagate` function

Given the grid and a cell coordinate, determines the type of the cell at the next tick of time (t+1).

`propagate` pseudocode

`propagate` code

1.e `update_grid` function

Takes the grid and the environmental parameters, returns a new, updated grid using the propagate function

`update_grid` pseudocode

`update_grid` code

1.f `simulate` function

the logic i would use is in the pseudocode, the actual function is implemented in a more complex way but the core logic is the same

`simulate` pseudocode

`simulate` code

1.g The Forest Fire Model

Run the following cell to simulate the forest fire over a 50 * 50 grid. See the resulting animation in the generated file `"forest_fire_animation.gif"`.

2.a

QUESTION:

Identify and explain the random variables involved in the `update_grid` function. How are these random variables utilized in the simulation?

ANSWER:

The random variable involved in updating the grid is actually used in the propagate function.

Each cell has one of the following types: BURNED, GRASS, DENSE_TREES, FOREST, BURNING or WATER.

Each one of these types has an arbitrary number between 0 and 1 associated to is: this number represents the likelihood *p* of the cell catching fire. i.e. water has 0, burned also has 0 because a burned patch of forest does not "reburn", dense_trees hase a higher number than forest etc...

Based on the *numbers* of burning neighbors I have created a factor, which is then multiplied to *p*, increasing it. This means: if I have a lot of burning neighbors, it will be more likely that I catch fire. Therefore, the probability that i catch fire depends on the status of my neighboring cells. As stated in the assignment: cells also have a certain probability *self_ignition* of self igniting. The value of this probability is arbitrarily chosen by me and it's small.

I generate a random number between 0 and 1 and there are two random variable functions that map in the following way: if *random_numer* < *p*, then the cell will be changed to burning. Otherwise, it will, if *random_number* < *self_ignition*, ignite by itself, or remain the same.

So the the first RV maps a randomly generated value to one of two outcomes: cell ignites because of neighbors or going through the second RV: cell ignites based on *self_ignition*.

2.b

QUESTION:

Describe the role of random number generators in the `update_grid` function. How are random numbers generated and utilized in the simulation?

ANSWER:

As decribed above, the random generator is used to determine if a cell will burn or not.

I also randomly generate the initial grid in the following way:

We want to select n random elements from a sequence of possible cell types with replacement (because the events are independent - meaning: if the previous cell is GRASS, this has no inpact on the next one being seomething in particular - at every cell-creation we consider all possible cell types).

This is done by generating a *random_number* between 0 and 1. This *random_number* will be used to select an element based on its weight, by accumulating the weights of all options and comparing them with the *random_number*. i.e. with weights = [0, 0.3, 0.2, 0.3, 0.1, 0.1], if my *random_number* is 0.6, it will be mapped to the cell type that corresponds to the third type because (0 + 0.3 + 0.2) < 0.6 < (0 + 0.3 + 0.2 + 0.3).

This is done by using random.choices() with weighted sampling (sampling with specified probabilities).

2.c

QUESTION:

Explain the stochastic processes involved in the `update_grid` function. How do these processes influence the behavior of the fire propagation?

ANSWER:

Stochastic elements are in both update_grid and propagate functions. They are introduced through the use of random numbers. These random numbers are utilized to determine whether a cell changes state (e.g., from unburned to burning) based on probabilities associated with the cell's type, its neighbors, and other factors. The propagate function determines whether a cell at position (x, y) in the grid should change its state (e.g., from GRASS to BURNING) based on probabilistic calculations: it checks the type of the current cell, computes the neighbors of the cell, and counts how many of those neighbors are in a burning state. It then calculates the probability of the cell catching fire, taking into account the type-specific ignition probability and the number of burning neighbors. Finally, it generates a random number to compare with the calculated probability to decide whether the cell ignites or remains the same. The outcome of this comparison determines whether the cell changes its state, adding a stochastic element to the simulation.

Random numbers are also used during the initial grid generation, where the random.choices() function is employed to randomly assign cell types with specified probabilities.

2.d

QUESTION:

Does `update_grid` exhibit Markov Chain characteristics? How many Markov chains are involved? Explain.

ANSWER:

The process of updating the grid in this fire simulation does exhibit Markov chain characteristics. A Markov chain is a stochastic process where the future state depends only on the current state and is independent of the past states, given the current state.

In this simulation: Each cell's state (e.g., BURNING, GRASS, etc.) at time step t depends only on its current state at time step t-1 and the states of its neighbors at time step t-1. The future state of each cell is determined probabilistically based on its current state and the states of its neighbors. These transition probabilities are determined independently for each cell in the grid based on its current state and the states of its neighbors. The simulation iteratively updates the grid over time, and the state of each cell at each time step depends only on the previous time step's state, adhering to the Markov property. Regarding the number of Markov chains involved, there is one Markov chain for each cell in the grid. Each cell undergoes state transitions independently based on its own characteristics and the states of its neighbors. Therefore, there are a total of grid_rows x grid_columns Markov chains in this simulation.

3.a

QUESTION:

Discuss the potential influence of wind speed and direction on fire propagation. In particular, propose modifications to the transition probabilities to include this factor.

ANSWER:

Wind direction could impact the propagation by increasing the probability that cells located in the given direction ignite, and the speed as well. Wind direction could be represented as a vector. Depending on the wind direction, cells upwind would have a lower chance of catching fire from a burning cell, while those downwind would have a higher chance. The higher the wind speed, the greater the probability that the fire will spread.

3.b

QUESTION:

Explore the effect of different vegetation types on fire spread. In particular, propose modifications to the transition probabilities to include this factor.

ANSWER:

I arbitrarily decided that some vegetations is at higher risk of ignition. It's explained in the questions above. They are represented with different colors:

```
CellType.GRASS: 'springgreen',
CellType.DENSE_TREES: 'green',
CellType.FOREST: 'forestgreen'
```

They also have different probabilities of igniting:

```
igniting_probabilities = {
    CellType.BURNED: 0,
    CellType.GRASS: 0.4,
    CellType.DENSE_TREES: 0.2,
    CellType.FOREST: 0.3,
    CellType.BURNING: 0,
    CellType.WATER: 0
    }
```

Grass has more probability of igniting due to the higher concentration of oxygen in fields and the type of vegetation being more delicate. Dense trees have less probability of igniting because of the water content of the trees and the lower oxygen and higher humidity. Regular forest is somwehere in between. I, a certified botanist, am 100% sure about the above information, therefore the simulation is 100% accurate :P

3.c

QUESTION:

Discuss how terrain elevation could affect fire spread. In particular, propose modifications to the transition probabilities to include this factor.

ANSWER:

Greater elevation means lower oxygen, which means less probability of the fire spreading. We could apply some scalar (mutiplicative factor) that reduces che probability all the cells igniting based on the elevation of the entire forest.

3.d

QUESTION:

Identify and discuss at least one other environmental or external factor that could affect forest fire propagation. In particular propose how this factor modifies the transition probabilities.

ANSWER:

The presence of water-neighbors means less probability of the fire spreading. We could apply some scalar (mutiplicative factor) that reduces the probability of a cell igniting based on the number of water neighbors. Propose how this factor modifies the transition probabilities.

3.e

QUESTION:

Implement the modifications discussed in the previous section to your Forest Fire Model. Ensure your implementation is able to simulate the fire propagation with the incorporated factors and demonstrate the working model. Explain any deviations from the initial ideas and the reasons behind such changes. Your code should compile and run correctly to demonstrate the modified model.

ANSWER:

The modifications are already implemented in the provided code and pseudocode :)