Two-dimensional propagation : forest fire

Guillaume Collet

Introduction

You're going to model a cellular automaton to study the spread of a forest fire. We'll be using a simple model so it won't be too complicated to implement.

Cellular Automaton : A cellular automaton is simply a matrix in which each cell takes a starting value and then changes values according to certain rules. It's called an cellular automaton because, once initialized, the cells automatically change value according to the rules.

You have to upload you python file when you think your project is finished.

Forest Fire Model

First, you need to define what's inside the cells of the matrix and associate a number to each state of cells.

Here, the matrix represents the forest, and each cell contains :
— a tree
— a burning tree
— ashes
— nothing

Then the rules of propagation are very simple :
— if a tree is next to a burning tree, it will burn on the next step.
— if a tree is burning, it becomes ashes on the next step.

You have to define how you will represent all these concepts in python. The only constraint is to use lists (and only lists).

1 Enter the Matrix

Now that you know what's inside the matrix, you need to create it in Python. Open a new file and name it `forest_fire.py`. First, you need to decide which parameters are important for generating the matrix.

1.1 Read parameters from the command line

In my opinion, the minimum parameters to create a matrix are :
— The width and height of the matrix.
— The density of trees (a percentage between 0 and 1).
— The number of burning trees.

Feel free to add more parameters if you think they are useful.

For example, if we want a 100x100 wide matrix (10000 cells) with a density of 67%, and 3 burning trees at start, we will launch our program like this :

```
$ python forest_fire.py 100 100 0.67 3
```

Write some lines of python to set the following variable names with the values from the command line. If you don't know how to get values from the command line, ask the "elders of the internet" about `sys` module.
— `width`, the width of the matrix
— `height`, the height of the matrix
— `density`, the density of trees
— `nb_burning`, the number of burning trees at start

Bonus : You may want to add flags to pass parameters, like this :

```
$ python forest_fire.py -w 100 -h 100 -d 0.67 -b 3
```

See modules `argparse` or `click` for more details.

1.2 Create the matrix

Now that you have the input parameters, you need to create the matrix. First, create a `list` of `list` of `int`, initialy set to 0. For example, if emptyness corresponds to 0 in your model, width=10 and height=5, the matrix should look like this.

```
[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
```

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]
```

Once created, you add trees given the density. You can see the density as a probability of appearance of a tree. You'll need module `random` to generate random numbers.

Then you can choose randomly the burning trees.

Example : In a model where empty=0, tree=1 and burning=2, with parameters width=10, height=5, density=0.5 and burning=1, the matrix should look like :

```
[[0, 0, 1, 1, 0, 1, 0, 1, 1, 0],
 [1, 0, 1, 0, 1, 1, 0, 1, 1, 0],
 [1, 1, 0, 1, 0, 0, 1, 0, 1, 0],
 [0, 0, 1, 0, 0, 1, 0, 1, 2, 1],
 [1, 1, 0, 0, 1, 1, 1, 0, 0, 0]]
```

1.3 Print the matrix

Instead of printing the lists, you want to print the matrix in a more readable way. I propose this visualisation but you can choose other values.

— empty cells '`.`'
— tree '`T`'
— burning tree '`f`'
— ashes '`_`'

each celle is separated by a space. During the printing process, you should not modify the matrix you created before. It's only a view of the matrix.

On the previous example, you should get :

```
. . T T . T . T T .
T . T . T T . T T .
T T . T . . T . T .
. . T . . T . T f T
T T . . T T T . . .
```

1.4 Simulate forest fire

To simulate the spread of the fire, you need to implement the rules you defined in your model. Each step, the fire spread to neighbors. So, given the previous matrix, you need to generate the next one and print it.

From the previous matrix, after one step you get something like :

```
. . T T . T . T T .
T . T . T T . T T .
T T . T . . T . f .
. . T . . T . f _ f
T T . . T T T . . .
```

And after another step :

```
. . T T . T . T T .
T . T . T T . T f .
T T . T . . T . _ .
. . T . . T . _ _ _
T T . . T T T . . .
```

1.5 Blowin' in the Wind

You may want to add the wind to change the direction of spreading... With wind direction and strength parameters. And some probabilities of spreading...