Computational Physics - Exercise 4

Bjarni Bragi Jónsson

October 2015

1 Task 1 - Code

I created the class Forest which has the attributes treeoccupationprobability, forestsize, forestgrid and lifetimeoffire. The class has several methods and furthermore uses the functions puttreesinforestgrid() and gettreeornothing(). The code can be seen below

```
class Forest(object):

    def __init__(self, tree_occupation_probability, forest_size):
        self.tree_occupation_probability = tree_occupation_probability
        self.forest_size = forest_size
        self.forest_grid = []
        self.create_forest_grid(tree_occupation_probability, forest_size)
        self.lifetime_of_fire = -1


    def create_forest_grid(self, p, N):
        empty_forest_grid = [[0 for x in range(N)] for x in range(N)]
        forest_grid = put_trees_in_forest_grid(empty_forest_grid, p, N)
        self.forest_grid = forest_grid

    def burn_first_cells(self):
        for i in range(0, self.forest_size):
            if self.forest_grid[0][i] == "tree":
                self.forest_grid[0][i] = 2


    def burn_forest(self):
        self.burn_first_cells()
        t=2
        while True:
            for line in range(0,self.forest_size):
                for col in range(0,self.forest_size):
                    if self.forest_grid[line][col] == t:
                        self.burn_neighbours(line, col, t)
            still_burning = self.is_it_still_burning(t)
            if not still_burning:
```

```
            self.lifetime_of_fire = t-1
            break
        t=t+1

def get_number_of_steps_in_shortest_path(self):
    step_list = []
    N = self.forest_size
    for col in range(N):
        element_in_last_line = self.forest_grid[N-1][col]
        if isinstance(element_in_last_line, (int, long, float)):
            step_list.append(element_in_last_line)
    if step_list!=[]:
        steps = min(step_list)-1
        return steps
    else:
        return "no percolation"


def is_it_still_burning(self, t):
   still_burning = False
   for line in range(0,self.forest_size):
       for col in range(0,self.forest_size):
           if self.forest_grid[line][col] == t+1:
               still_burning = True
   return still_burning




def burn_neighbours(self, line, col,t):
    self.burn_east(line, col,t)
    self.burn_west(line, col,t)
    self.burn_south(line, col,t )
    self.burn_north(line, col,t )

def burn_east(self,line, col, t):
    N = self.forest_size
    if col <= N-2:
        if self.forest_grid[line][col+1]=="tree":
            self.forest_grid[line][col+1] = t+1

def burn_west(self,line, col, t):
    N = self.forest_size
    if col>0:
        if self.forest_grid[line][col-1]=="tree":
            self.forest_grid[line][col-1] = t+1

def burn_south(self,line, col, t):
    N = self.forest_size
    if line <= N-2:
```

```
            if self.forest_grid[line+1][col]=="tree":
                self.forest_grid[line+1][col] = t+1

    def burn_north(self,line, col, t):
        N = self.forest_size
        if line > 0:
            if self.forest_grid[line-1][col]=="tree":
                self.forest_grid[line-1][col] = t+1
```

```
def put_trees_in_forest_grid(empty_forest_grid, p, N):
        for line in range(N):
            for col in range(N):
                new_element = get_tree_or_nothing(p)
                empty_forest_grid[line][col] = get_tree_or_nothing(p)
        forest_grid = empty_forest_grid
        return forest_grid


def get_tree_or_nothing(p):
        rnd = random.random()
        if rnd<p:
            return "tree"
        else:
            return "empty"
```

2 Task 2 Forest Fires

I torched several forests - the green color resembles trees, white resembles an empty spot and the red is for burning trees. The results can be seen below:

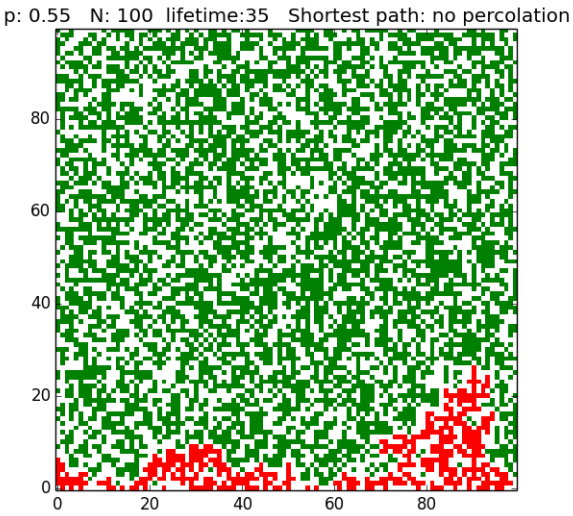

Figure 1: p=0.55

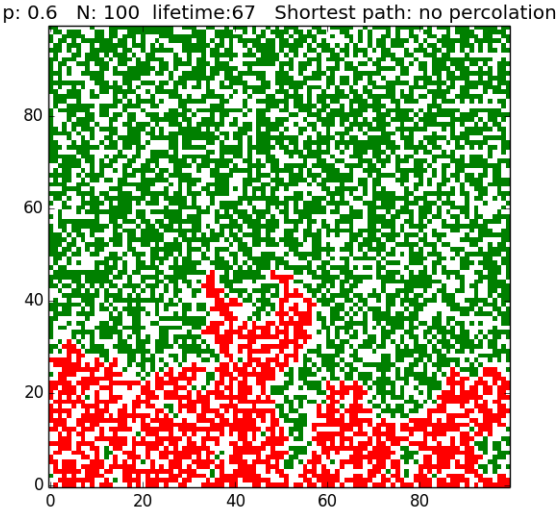

Figure 2: p=0.6

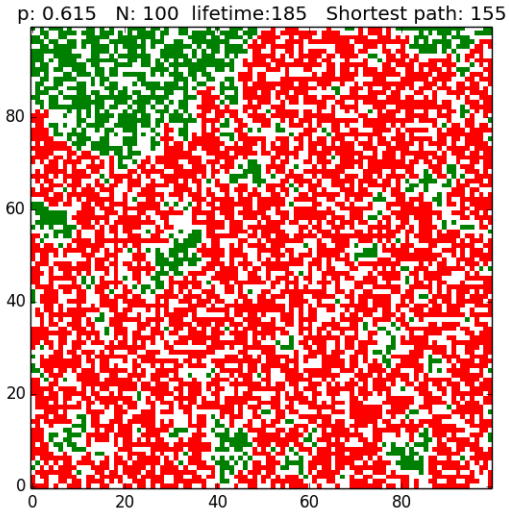

Figure 3: p=0.615

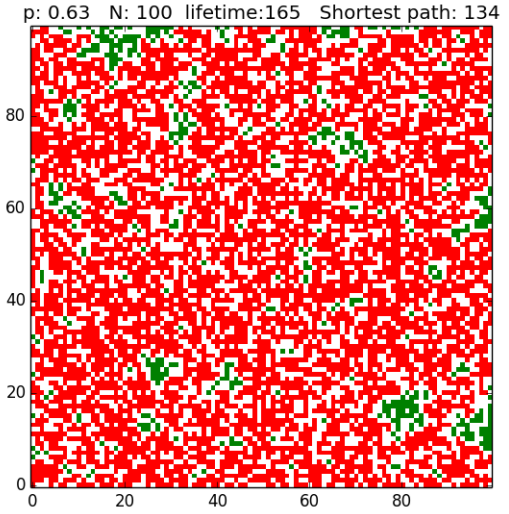

Figure 4: p=0.63

3 Task 3 threshold probability

3.1 fixed n - fixed p

For N=30, p=0.6 I created 1000 trees. the fraction of samples with percolative cluster was 0.577 and the the average shortest path (when percolation occured) was 43.

3.2 Varying N and varying p

On figure 4 we can see that percolation thresh- old is approximately 0.59.

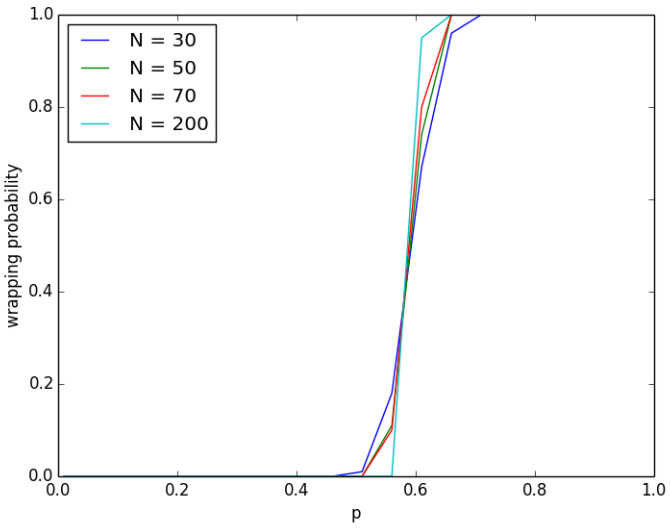

Figure 5: Wrapping probability as a function of the tree occupation probability for different forest sizes