Project – Simulating a Forest Fire

Deadline: Week 13

The project is an assignment that has to be completed in the groups specified at NTULearn BS1009 page. The purpose of the project is to solve a computational problem as a team. To this end, you will use your newly gained Python programming skills to implement a **Forest Fire** simulation. Furthermore, you will use a **Monte Carlo simulation** that uses randomness to identify a forest that is most resistant to fire. The project starts in **week 9** and is divided into 4 tasks:

Task 1) Week 9: Making a random forest, worth 5 points.

Task 2) Week 10+11: Setting the forest on fire (5 points)

Task 3) Week 12: Using Monte Carlo simulation to find a forest density that increases the survivability of trees (5 points)

Task 4) Week 13: A project report (5 points), comprising 2 pages describing the scripts, your approach and conclusion is then submitted to “Project: Forest Fire” on NTULearn.

The schedule of the project is given below (Figure 1).

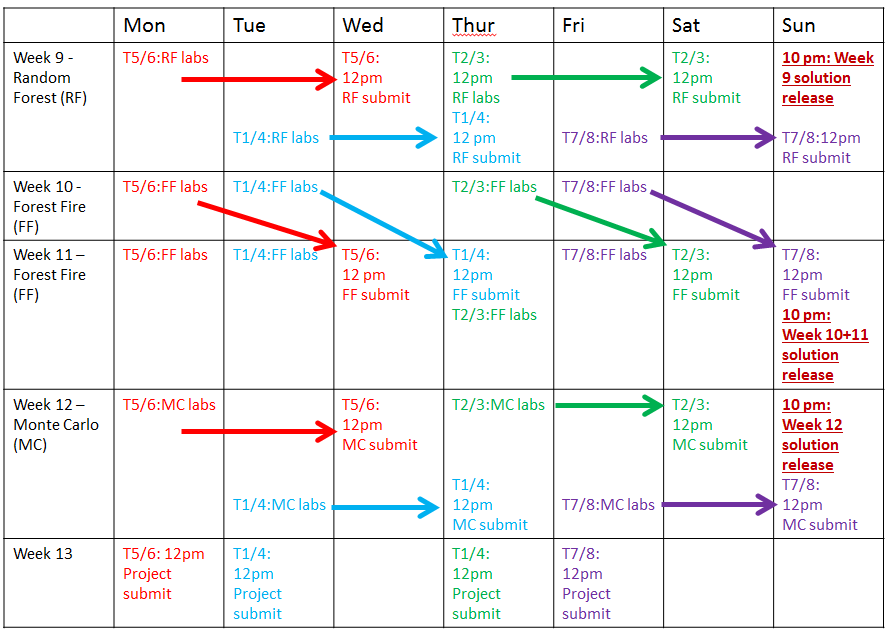


Figure 1. Project schedule for weeks 9-13. The project topics (random forest, forest fire, Monte Carlo) are shown in the first column. The practicals and submission days (2 days after a practical) for the different tutorial groups (Monday: red, Tuesday: blue, Thursday: purple, Friday: green) are indicated. The release time of a solution for a given week is indicated by red underlined text.

Starting from week 9, your group has to submit a mini-report with the script containing your solution (with sufficient/appropriate comments) to a given task (Figure 1). The solution has to be submitted by one person to “Project: Forest Fire” on NTULearn. The solution should be presented as a text document named GroupName\_Week. For example, if snakesss submit their mini-report for week 12, the file must be called snakesss\_12.docx.

The deadline for the submissions are:

1. Week 9 (random forest): two days after your schedule class, 12 pm (Figure 1). For example, T5/6 group (Monday) has to submit their solution to a random forest task (week 9, Monday lab session) by Wednesday 12 pm (week 9), latest.
2. Week 10-11 (forest fire): The forest fire simulation is more complex and you will be given nine days to complete this task. For example T7/8 (Friday) needs to submit their mini-report on week 11 Sunday, 2pm.
3. Week 12 (MC simulation): two days after your schedule class, 12 pm (Figure 1), similarly to week 9.
4. Week 13 (final report): As shown in Figure 1.

Late submissions are ignored and will result in 0 points for that week’s task.

To ensure that nobody gets stuck at a particular point of the project, we will release a solution for the given week by Sunday 10pm. For example, if you get stuck implementing the forest fire (week 10+11), we will release a solution on Week 11 Sunday, at 10pm that you can use to continue with of the project. However, note that using the given solution does not give points when the project is evaluated at the end of the course. For example, if you solve the random forest (5 points), forest fire (5 points) and project report (5 points) tasks, but used the given solution to solve the Monte Carlo simulation, you will get 15 out of 20 points.

While the project is done in a group, each group member is evaluated based on the completed functionalities of the program and peer evaluation (see the rubric at the end of the document).

# Project description

In this project, you will learn how to write a simulation of a natural disaster: a forest fire. You will then use random numbers to identify a forest density that results in the highest number of trees that survive a fire.

An example of a simulated forest fire can be seen [here](https://youtu.be/WucU8AyUjeQ). The forest consists of flammable trees (green squares) and inflammable ponds (blue squares). A fire (red square) starts in the middle of the forest, and for each time unit, it spreads to neighboring trees. A burning tree turns to ashes (black square) after burning for one time unit. At some point, the forest stops burning, as the fire has consumed all available trees. Note that ponds are not flammable and are able to insulate trees from catching fire. A forestFire\_template.py file, containing the template for the script can be downloaded from BS1009 NTULearn course site.

# Task 1 (Week 9): Making a random forest

**Description**: you need to use abstraction to represent a forest. Our simple abstraction will assume that the forest: (i) is rectangular, with a well-defined width and height, (ii) has tree density, defined as: and (iii) contains either trees or ponds.

**Task 1.1**: Make a function createForest that as arguments takes width, height, and density, and returns a forest matrix (which is effectively a list of lists; see Figure 2A for its syntactic representation to understand why) that contains the defined number of rows and columns (Figure 2A shows a matrix of 6 rows and 5 columns). The matrix is thus an abstraction of a forest, where 1s and 0s represent trees and ponds (Figure 2B shows a corresponding image of the matrix). To generate a forest with a randomized placement of trees, use Python’s [random](https://docs.python.org/2/library/random.html) module, as it contains many useful functions that can generate random numbers and randomness in your data. For example, you can use random.shuffle() function to randomize the order (shuffle) of trees in your forest. Some examples of random forests at different densities are exemplified in Figure 2C.

|  |  |
| --- | --- |
|  | Figure 2. Making and visualizing a random forest. A) A list of lists (matrix) containing six rows and five columns. 0 and 1 correspond to a pond and a tree, respectively. B) A graphical representation of the forest. The blue and green squares represent ponds and trees, respectively. C) Examples of random forest for tree densities of 0, 0.1, 0.5, 0.9 and 1. |

# Task 2 (week 10+11): Setting the forest on fire

**Description**: after creating a random forest that contains trees (encoded by 1s) and ponds (encoded by 0s), we start a fire (encoded by 3) in the middle of the forest (Figure 3A). After one time unit (i.e. one iteration of the loop), the fire spreads to a neighbouring trees, but it does not spread to ponds, ashes (encoded by 2s) or other fires. This means that ponds are able to insulate certain regions of the forest (Figure 3C and 3D demonstrates how ponds can insulate trees). Furthermore, a fire (3) turns into ash (2) in the next iteration of the loop.

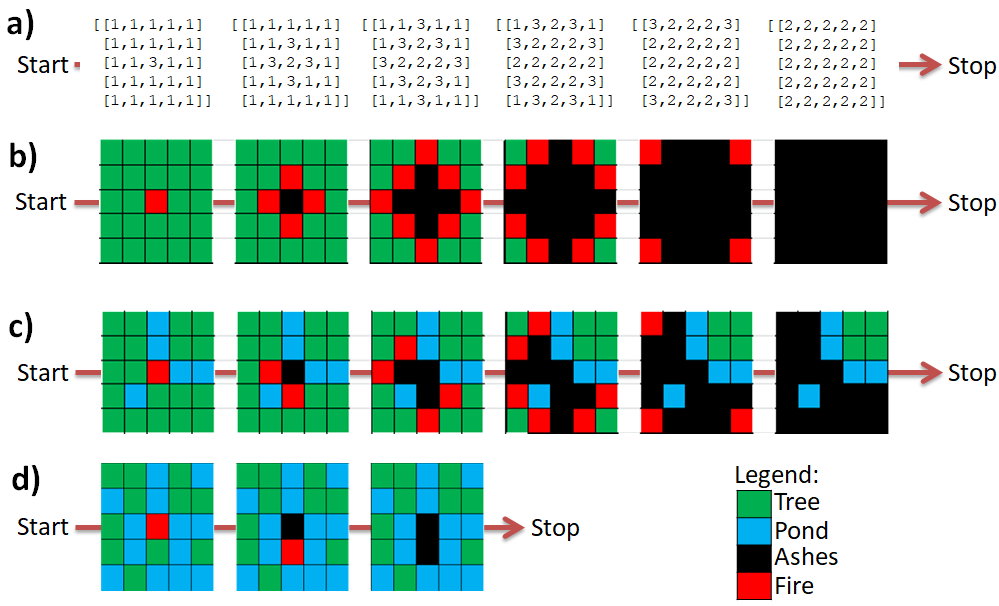


Figure 3. Setting the forest on fire. A) An example of fire spreading in a forest of density 1. The fire (3) start in the 3rd row, 3rd column on time = 0, and spreads to the north, west, east and south (time = 1). The burning tree at time = 0 is reduced to ashes (2) at time = 1. Similarly, at time = 2, the fire spreads to neighbouring trees, and trees that were burning at time = 1 are reduced to ashes. The forest stops burning at time = 5, when all trees have burned down. B) A graphical representation of the forest fire shown in A). C) An example of a forest fire of density = 0.83 (25 out of 30 cells contain trees), where blue squares represent ponds. The fire burns for six time units and stops when all available trees have burned down. Note that the pond squares have insulated four trees from the fire. D) An example of a forest fire in a random forest with density = 0.4 (12 out of 30 squares are trees). The forest stops burning after 3 time units as the fire is contained by the ponds.

**Task 2.1**: Make a function fireSpreading that takes a forest matrix containing the trees, ponds, fire and ashes, and returns the forest matrix where the fire has spread for one time unit. For example, assume that we have a simple forest of 1 row and 3 columns:

forest = [[1,3,1]]

Passing the forest in the fireSpreading should return a forest where the time has advanced by one unit.

forest = fireSpreading(forest) #forest becomes [[3,2,3]]

Passing the forest again advances the time by one unit.

forest = fireSpreading(forest) #forest becomes [[2,2,2]]

**Task 2.2**: Write a loop that passes the forest to the fireSpreading() function for as long as there is a fire in the forest. For example, since the forest represented by [[2,2,2]] does not contain any fires (only ashes), the fire simulation can terminate.

# Task 3 (week 12): Monte Carlo simulation – finding the fireproof forest

**Description**: After the fire is nicely spreading, it is time to use the model for something useful: predicting which forest density results in a forest with the highest number of surviving trees. For this, we will use a [Monte Carlo method](https://en.wikipedia.org/wiki/Monte_Carlo_method), that used repeated random sampling to obtain a numerical result. Monte Carlo method (MCM) was developed in late 40s by Stanislaw Ulam while he was working on nuclear weapons project at Los Alamos National Laboratory. MCM is now widely used in physics, economics, math and computational biology.

|  |  |
| --- | --- |
|  | Figure 4. Estimating the optimal forest density with the Monte Carlo approach. A) Five random forest fires of density 0.5. The number 9, 3, 2, 2, 14 indicate the number of surviving trees. B) Fire random forest fires of density 0.8. The numbers 1, 0, 0, 0, 0 indicate the surviving trees. |

The essence of MCM is shown in Figure 4, where five random forests of density 0.5 (Figure 4A) and five random forests of density 0.8 (Figure 4B) have been set on fire. As you can see, forest of density 0.5 tends to have more surviving trees than a more dense forest. Furthermore, you can see that for each random forest, a different configuration of the ponds results in a different number of surviving trees. Therefore, to be able confidently state that a given forest density is better at keeping trees alive, you need to run at least 100 forest fires for a given density, and compare the results of the runs.

**Task 1**: Make a loop that iterates over forest densities of 0, 0.1, 0.2, … , 1. For each iteration, generate and set fire to 100 forests and record the number of surviving trees.

# Task 4 (week 13): Final report

**Description**: Write a report that summarizes what you have learned and the interpretation of the results. The report comprises:

**Task 4.1:** An introduction describing the aim and background of the project (0.5 page).

**Task 4.2:** A description of your script, more specifically, the functions and loops (0.5 page).

**Task 4.3:** Interpretation of the MCM analysis. Which forest density results in the highest number of surviving trees and why? What other applications could you envision using MCM for? Please provide examples (0.5 page).

**Task 4.4:** Concluding remarks. What did you learn from this project, how can you apply the gained knowledge and how would you improve the course (0.5 page)?

**Final Project assessment rubric**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Standards | | |
| Fail standard (0-39%) | Pass standard (40-80%) | High standard (81-100%) |
| Criteria | Demonstrated less than 40% of the functionalities according to the specifications; Python and pseudocode accepted.  OR  Peer Evaluation: Inadequate contribution to team learning, i.e. a team member did not do his/her fair share of work. Average peer score <=2.5 | Demonstrated 40% to 80% of the functionalities according to the specifications; Python and pseudocode accepted.  AND  Contribution to team learning was at least adequate, i.e. a team member put in his/her fair share of work. Average peer score >2.5 | Demonstrated more than 80% of the functionalities according to the specifications. Only Python code accepted.  AND  Contribution to team learning was significant, i.e. not only did a team member put in his/her fair share of work, but also supported other members’ learning. Average peer score > 3 |

**Peer Evaluation Form (This is only an example. Please use the online weblink that will be provided for submitting your peer evaluation)**

Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Write the names of your group members in the numbered boxes. Then, assign yourself a number for each listed attribute. Finally, do the same for each of your group members and total up the values.

Values: 1=Strongly Disagree; 2=Disagree; 3=Agree; 4=Strongly Agree

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | |  | Group Members | |  |  |
| Attribute | Yourself | | Member #1: | Member #2: | Member #3: | Member #4: |
| Contributed the fair share of work |  | |  |  |  |  |
| Contributed to one another’s learning |  | |  |  |  |  |
| Total Values |  | |  |  |  |  |