Workshop

Computational Thinking and Programming 24/25

Silvio Peroni

Digital Humanities Advanced Research Centre (/DH.arc)
Department of Classical Philology and Italian Studies, University of Bologna, Bologna, Italy
silvio.peroni@unibo.it – @essepuntato – https://www.unibo.it/sitoweb/silvio.peroni/en

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA
DEPARTMENT OF CLASSICAL PHILOLOGY
AND ITALIAN STUDIES

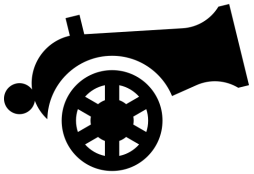
The Cannon Escape

Plot

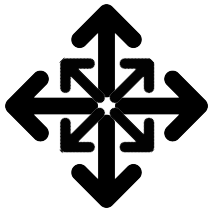
What happened in the previous episodes: Myntrakor, also known as Who Must Not Be Thought, created the Tower Labyrinth, a complex system made of 100 squared mazes of different dimensions placed one upon the other designed to custody the Book of Indefinite Pages, containing all the possible books ever written in just one portable volume. Myntrakor hired Urg, a despicable man, as a guarantor of the Tower Labyrinth. Urg implemented a mechanism that allows the walls to move while someone is trying to reach the exit, making it impossible for thieves to enter the Tower Labyrinth, steal the Book of Indefinite Pages, and go out from all the mazes alive.

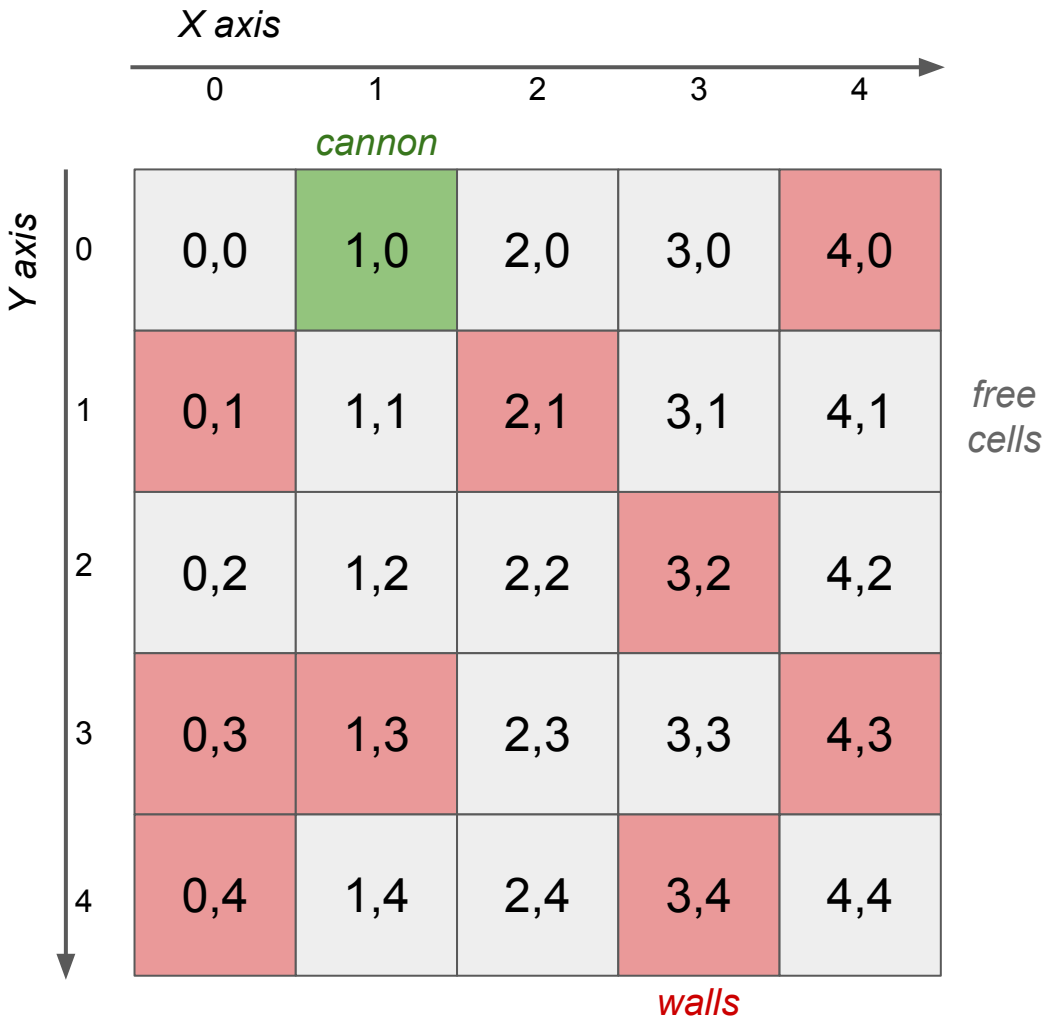
However, the Interplanetary Guild of Thieves created a special cannon that can be quickly teleported into a maze and can destroy the walls of each maze of the Tower Labyrinth. The problem is that the cannon appears in different positions in each maze, and a thief, to be used to exit the maze, must destroy as many walls as possible with one shoot before moving to the next maze using the free corridor created by the shooting. Can the Interplanetary Guild of Thieves steal the Book of Indefinite Pages and crash all the mazes of the Tower Labyrinth, finally collapsing the entire evil structure created by Myntrakor?

Structure of a maze

The cannon 

shoots just one bullet per maze in one of the following directions





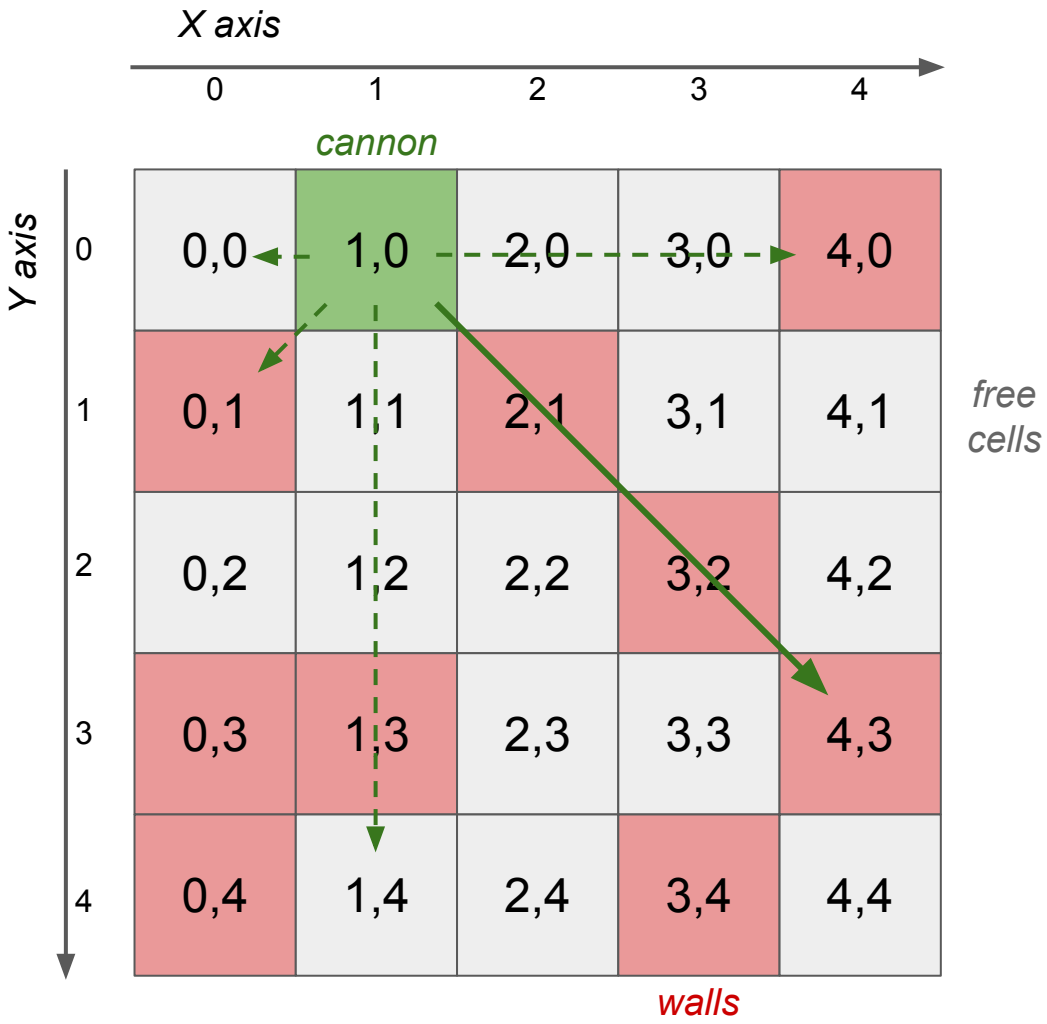
Rules

1. Each room is a square which is initially **filled with an arbitrary number** of free cells and walls
2. The thief **controls the cannon**, initially positioned in a precise cell
3. The thief has just **one shot** of the cannon to use per room that should be in one of the eight possible direction (up, down, left, right, and the four diagonals)
4. The cannon **cannot** move in another cell, and its bullet destroy everything is the trajectory between the cannon and the borders of the maze
5. The thief can escape if, with a cannon shot, (s)he can to **destroy the maximum number of walls** considering the position in which the cannon has been placed
6. The thief **cannot cheat** – and (s)he is caught to cheat if:
    a. (s)he claims that all possible trajectories for the cannon shooting do not contain any wall
    b. (s)he claims to have destroyed a wall that did not exist in the maze

Example

Considering the configuration of the maze and the position of the cannon, there are only five possible direction for the shoot, indicated with green arrow

The shoot indicated by the arrow with the solid line is the one that allows the thief to destroy the maximum number of walls



Function to implement

```
def do_move(maze, cannon)
```

It takes in input:

- **`maze`**, a list of lists of dictionaries representing the rows of the maze and the cells in each row
- **`cannon`**, a tuple of X/Y coordinates identifying the current position of the cannon

It returns a list of tuples of two items, where each tuple defines the X/Y coordinates of the walls that have been destroyed shooting with the cannon – e.g. `[(2,1), (3,2), (4,3)]`

Example of a board

```
[           The main list defining the maze
    [           The first row (i.e. a list) of the maze
        {   'x': 0,             X coordinate of the first cell of the first row
            'y': 0,             Y coordinate of the first cell of the first row
            'type': 'free' },   type of cell ('free' or 'wall')
        {   'x': 1,             X coordinate of the second cell of the first row
            'y': 0,             Y coordinate of the second cell of the first row
            'type': 'wall' }, ...   type of cell ('free' or 'wall')
    ],
    [ … ], …  The second row (i.e. a list) of the maze
]
```

How it works

The file responsible for showing how to run the game is **`run.py`**

The initial part of the file **`run.py`** contains an exemplar 4x4 board with free cells and walls, followed by the position of the cannon

The function **`do_move`** is included in the file **`group.py`** and will be called in the body of the while loop in **`run.py`** – please note that the code in **`run.py`** does not check if the move is valid or not

To run your implementation of **`do_move`** you can run
`python run.py`

(The initial implementation of **`do_move`** will return a list with one cell which is outside the maze)

Points to the groups

1 point will be assigned to all groups that developed a function that allowed their thief **to avoid cheating,** i.e. in case no walls were destroyed or (s)he pretended destroying a wall which did not exist

1 point will be assigned to the group that **missed the minimum number of walls** overall, out of 100 mazes

1 point will be assigned to all groups that developed a function that allowed their thief to successfully escape of **at least 30** different mazes

1 point will be assigned to all groups that developed a function that allowed their thief to successfully escape of **at least 90** different mazes

Material and submission

All the files are available at
https://github.com/comp-think/2024-2025/tree/main/docs/workshop

At the end, each group must send an email to silvio.peroni@unibo.it where are specified:

- The name of the group
- The members of the group
- The file containing the implementation of `do_move` (rename it from `.py` to `.txt` before attaching it to the email)

End

Workshop

Computational Thinking and Programming 24/25

Silvio Peroni

Digital Humanities Advanced Research Centre (/DH.arc)
Department of Classical Philology and Italian Studies, University of Bologna, Bologna, Italy
silvio.peroni@unibo.it – @essepuntato – https://www.unibo.it/sitoweb/silvio.peroni/en

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA
DEPARTMENT OF CLASSICAL PHILOLOGY
AND ITALIAN STUDIES