Homework 3 - Stochastic Task

Introduction

In this exercise, you continue controlling the wizards from HW1. However, this time the death eaters' movement and the horcruxes are not deterministic.

Task

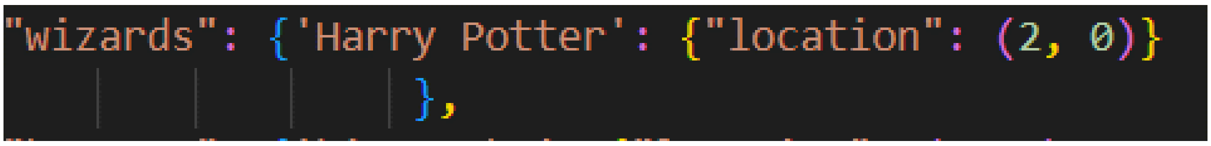
The environment is like the one in HW1: The same grid world, with the same wizards, horcruxes and death eaters. Key differences include:

- The death eaters do not move exactly as their path states (more on that below).
- The execution has a limited number of turns.
- The goal is to collect as many points as possible (more on that below).
- When encountering the death eaters, the wizards pay a fee (more on that in the 'points' section)

Wizards:

Now, the input of the wizards is a little different than in HW1:

- The "location" parameter is the location of the wizard.

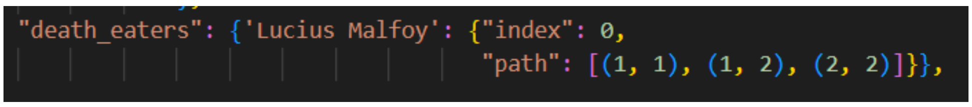


```
"wizards": {'Harry Potter': {"location": (2, 0)}
                            },
```

In this HW, the wizards have infinite lives but encountering a death eater will result in a penalty (more on that in the Points section)

The Stochastic Death Eaters

Each death eater has a certain path, the same way as in HW1. However, in this exercise they do not have to precisely follow it: Each death eater can stay, move forward, or move backwards in the path. For example, it the path is $A \rightarrow B \rightarrow C \rightarrow D$ and the death eater is in tile B, he can move to tile A, C or stay in tile B. The probabilities are equal for every direction.
If the death eater was in tile A, he can move to tile B or stay in tile A. The probabilities are also equal for every direction.
The input for the death eaters is:



```
"death_eaters": {'Lucius Malfoy': {"index": 0,
                                   "path": [(1, 1), (1, 2), (2, 2)]}},
```

The "path" argument is the same path as was in HW1.
The "index" indicates where the death eater is on his path.
For example: if we are at (1,2) (i.e index = 1), then the probability to move to (1, 1) is 1/3 and the probability to move to (2, 2) is also 1/3, and the probability to stay in (1, 2) is 1/3 as well.
If the death eater is at (1, 1), then the probability to move to (1, 2) is 1/2 and the probability to stay in (1,1) is also 1/2 .
Notice:

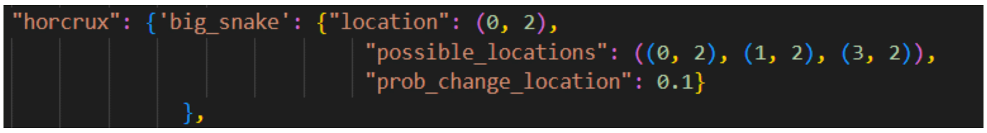
- The path is not necessarily simple so multiple indices can have the same location. The index can increase, decrease, or stay the same as explained above.

Lord Voldemort:

He is not in this HW. Hence, there isn't an action named Kill or a cell marked with V.

The stochastic Horcruxes

As you have probably realized, we are dealing with special and magical horcruxes. As such, they can change their location at random at each turn. For example, let "big_snake" be a horcrux with the following parameters:



```
"horcrux": {'big_snake': {"location": (0, 2),
                          "possible_locations": ((0, 2), (1, 2), (3, 2)),
                          "prob_change_location": 0.1}
           },
```

Here, the horcrux's name is "big_snake", its location is (0, 2), each turn there is a 10% chance that the horcrux will draw a location, uniformly at random, from the "possible_locations". Notice, the new location **can** be the location he is currently at and thus, will not move.
Each location in the "possible_locations" must be reachable.
Important: if a wizard destroys a horcrux in the same turn it changed its location, the horcrux is still destroyed.

Actions:

The action syntax and rules remain as they were in HW1, with 2 new actions:

1. **Reset**: It resets the places of wizards, death eaters and to the initial state. The action does not reset the number of turns or the points accumulated. The syntax is simple – "reset" (Just a string, not a tuple containing the string).
2. **Terminate**: This action terminates the game. This may be of use if resetting yields negative expected results. After you terminate the game, the game is over, and no more actions can be made. The syntax is simple – "terminate" (Just a string, not a tuple containing the string).

The rest of the actions stay the same – a tuple of "atomic actions". Notice that when the action is Reset or Terminate – it resets or terminates the entire game, i.e. not for one ship alone. Also, for this exercise every horcrux has a name, and thus the destroy action syntax is modified to ("destroy", wizard_name, horcrux name) rather than ("destroy", wizard_name, horcrux_index).

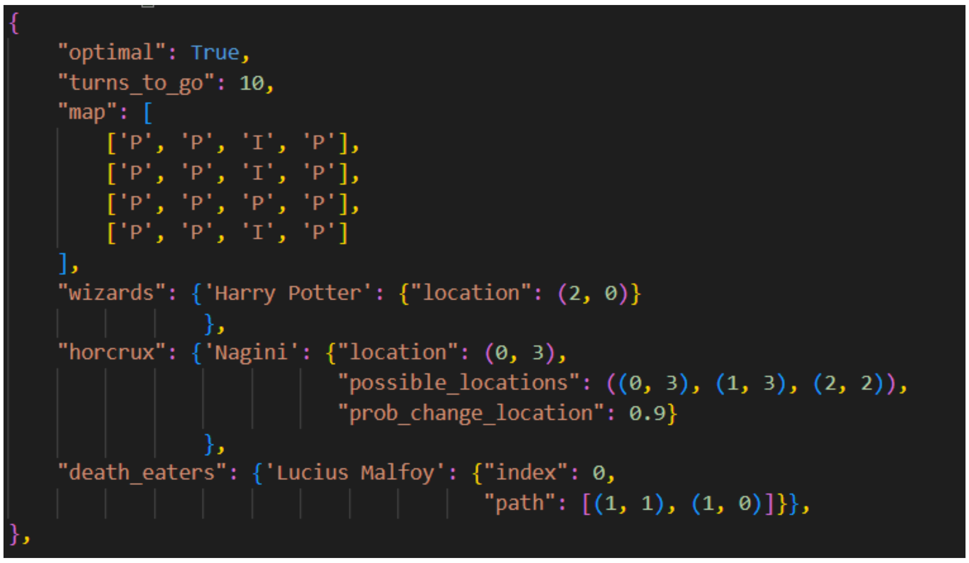
Points:

In this exercise, your goal is to achieve maximum points. Points are given for the following:

- Successfully destroying a horcrux: 2 points.
- Resetting the environment: -2 points.
- Encountering a death eater -1 points for each wizard that encounters a death eater each turn. For example, if 2 wizards encounter a death eater in some turn t, then you get -2 points. If 2 death eaters encounter 2 wizards in some turn t (all four are at the same spot), then you get -4 points.

The input

The input is presented as follows:



```
{
    "optimal": True,
    "turns_to_go": 10,
    "map": [
        ['P', 'P', 'I', 'P'],
        ['P', 'P', 'I', 'P'],
        ['P', 'P', 'P', 'P'],
        ['P', 'P', 'I', 'P']
    ],
    "wizards": {'Harry Potter': {"location": (2, 0)}
                },
    "horcrux": {'Nagini': {"location": (0, 3),
                           "possible_locations": ((0, 3), (1, 3), (2, 2)),
                           "prob_change_location": 0.9}
                },
    "death_eaters": {'Lucius Malfoy': {"index": 0,
                                       "path": [(1, 1), (1, 0)]}},
},
```

- "optimal" indicates that the agent must act optimally.
- "map", "wizards", and "death eaters" are as explained above.
- "horcrux" is as described above also.
- "turns_to_go" – the number of turns the execution takes.

The task

Code-wise, your task is to implement two agents (WizardAgent and OptimalWizardAgent). Each agent shall have (at least) two functions:

- __init__(self, inital) – A constructor. Must finish in 300 seconds.
- act(self, state) – A function that returns an action given a state. Must finish in 5 seconds.

Code handout

Code that you receive has 5 files:

1. ex3.py - The only file you should modify, which implements your agent.
2. check.py - The file that implements the environment, the file that you should run.
3. utils.py - The file that contains some utility functions. You may use the contents of this file as you see fit.
4. inputs.py: self-explanatory.

Note: We do not provide any means to check whether the solution your code provided is correct, so it is your responsibility to validate your solutions.

Submission and grading

You are to submit only the file named **ex3.py** as a python file (no zip, rar, etc.). We will run check.py with our inputs and your ex3.py. The check is fully automated, so it is important to be careful with the names of functions and classes.

The grades will be assigned as follows:

- 60% - If your WizardAgent finishes solving the test inputs with a strictly positive score.
- 25% - If your OptimaWizardAgent solves all given problems optimally.
- 25% - Competitive Part: Grading on the relative performance of the WizardAgent, based on points.

- There is a possibility to earn bonus points by reporting bugs in the checking code. The bonus will be distributed to the first reporter.
- We will use Python 3.10 for testing your submission. You may only use the modules in https://docs.python.org/3.10/py-modindex.html.
- The submission is due on 4.2.2025 at 23:59.
- Submission in pairs/singles only
- Write your ID numbers as strings in the appropriate field ('ids' in ex3.py). If you submit alone, leave only one string.
- The name of the submitted file should be "ex3.py".

Important notes:

- You are free to add your functions and classes as needed. Keep them in the ex3.py file.
- We encourage you to double-check the syntax of the actions as the check is automated.