Introduction to AI

Project : Pac Man

Thanks to Berkeley University

Samy Aittahar

25th November 2017

1 Pratical Information

- Deadline: December 22.
- You must work in groups of 2 students.
- Deliverable 1: A Python implementation of the classes `Agentsearch` and `Agentghost`, as defined in the `agentsearch.py` and `agentghost.py` templates. See source code for more details
  - ⚠If your agent's action violates the game constraints, it will be ignored,
  - Free to use any library, as long as it does not hurt your code organization and clarity, and you are able to explain the principles behind it.
  - The name of your agent classes should be `Agentsearch`*numstudent1 numstudent2* and `Agentghost`*numstudent1 numstudent2* where *numstudent1* and *numstudent2* are your student IDs without the "s". Name of your main agent class file should be respectively `agentsearch`*numstudent1 numstudent2*`.py` and `agentghost`*numstudent1 numstudent2*`.py`.
- Deliverable 2: A report of 7 pages max. in which should appear, for each stage of the project :
  - The approaches you have considered to implement your agents.
  - Their limitations for the current and the next stage of the project.
  - Comparisons with some naive approaches.

  - Briefly but clearly discuss about better strategies than your current work. "We could use X" is not enough, you need to explain the benefits (and drawbacks) of the claimed "better" approaches.

- Upload your deliverables (`agent.py` and `report.pdf`) as a tar.gz archive on the Montefiore submission platform.

- You will be evaluated on the following criterions :
  - Score + computation time of your agents.
  - Organization and clarity of your code.
  - Quality of your report.

2 PAC-MAN GAME

The games consists in eating all food dots while avoiding ghosts in a fully observable maze. Game ends when eitehr Pacman has eaten all the food dots (winning game) or has been hit by a ghost (game over). Figure 2.1 shows an example of a Pacman maze with a single food dot.

2.1 STEP 1 : SEARCH

Pacman is let alone in a maze with an arbitrarily distribution of food dots over the map. The goal is to eat them all while minimizing a score based on the time spent to eat the food dots.

You are encouraged to try several algorithms as described in the *Search* lecture. Remember there is an implicit time limit, as the game engine does not wait your computation to go on.
Complete the `Agentsearch` class template to address this step. More details can be found on the source code for completion.

2.2 STEP 2 : SEARCH + GHOSTS

Ghosts (one or more) are added to the maze. Pacman has to avoid them while collecting all the food dots. If Pacman collides with a ghost, the game ends there with a negative score. All ghosts agents will behave using one of the following patterns, known by Pacman in game (except the last one) :

- Pattern 1 : Counterclockwise left (see Figure 2.2 for more details)
- Pattern 2 : Always move greedily towards Pacman
- Pattern 3 : Semi-random pattern defined below.
  - Pattern 2 : 50%

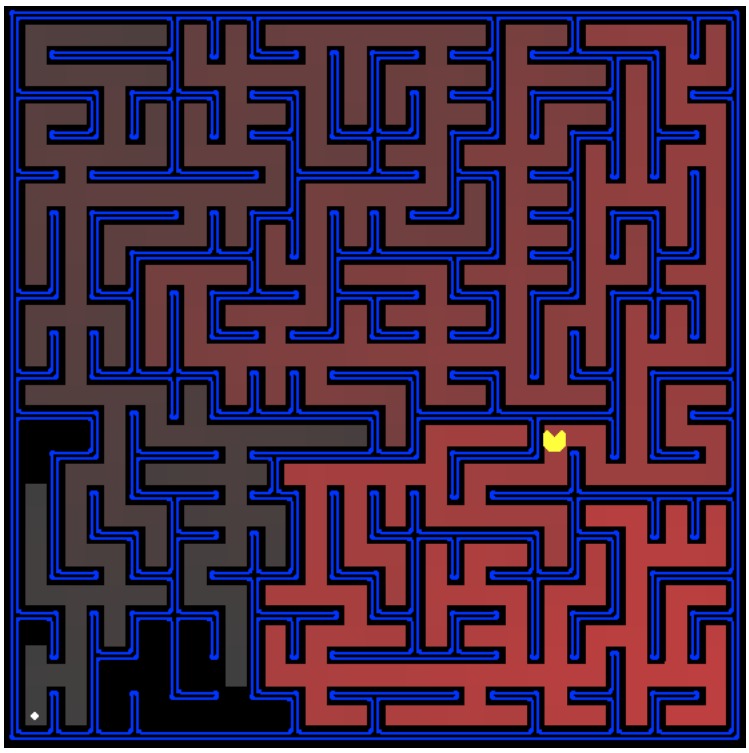

Figure 2.1: Pac man and food dots in a maze

  - Pattern 1 : 25%
  - Random : 25%

- Pattern 4 (unknown by Pacman) : either pattern 1, 2 or 3.

Big food dots are also added. When Pacman eats one, he has the ability to eat the ghosts for a fixed amount of time, which will be provided to the Agentghost class. An eaten ghost "repops" immediately in its initial position.

Like in the previous step, you are encouraged to compare several approaches described in lectures or even coming from other sources. In the latter case, you'll have to explain carefully the principles behind the chosen approaches.

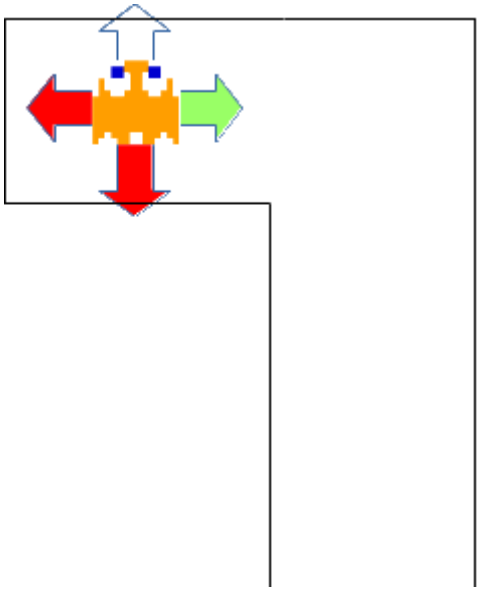

Figure 2.2: Counterclockwise left. Ghost try first to go left and if it is not possible, analyze moves in a counterclockwise order until a legal move is found