Introduction to AI

Search problem : Pac-Man

Project Part 1/3

Samy Aittahar

1 Practical Information

- Due to Oct 01, 2018.
- You must work in **groups of 2 students**.
- Github link : <To be defined>

2 Assignment

In this game, Pac-man navigates through a maze filled of foods. The goal is to accumulate points by eating all the foods. The maze is not necessarily fully filled of them. Your task is to implement the best possible agent which collects all the dots while minimizing the time spent on the maze. More precisely, you need to implement the following algorithms : (i) depth-first search, (ii) breadth-first search, (iii) uniform-cost search, (iv) A* with closest dot heuristic and (v) A* with your own modifications (e.g. heuristic, data structures...). The latter needs to be better than the others baseline algorithms, and to be able to solve a maze with and without a computational time budget during the game.

3 Deliverables

All the following deliverables need to be enclosed together in an archive. See the corresponding project section in the submission platform[1] for more details.

[1] https://submit.montefiore.ulg.ac.be/

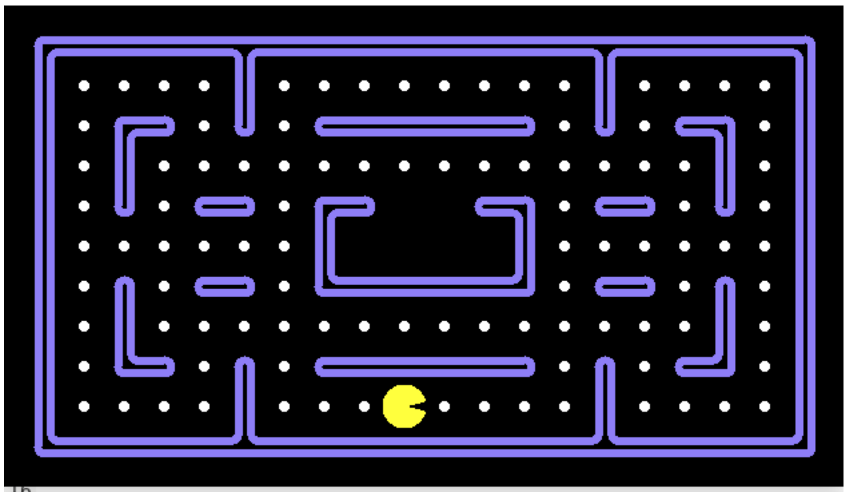

Figure 2.1: Pac man and food dots in a maze

3.1 Implementation

You need to send, at least, a Python 3 source code file, containing the class template[2] to complete on your own. An additional class example, involving a random agent, is also available[3] . Any additional source code file that your class implementation includes needs to be enclosed in a subfolder named *modules*. Your source code needs to be formatted in order to follow the PEP-8[4] style conventions. Make sure to properly follow the instructions contained in the class template.

3.2 Report

You need to provide a PDF report of 4 pages max. Your report must (i) describe your approach and justify the improvements over the baselines seen in lectures that are relevant to the pacman game, (ii) to compare the performance of your agent over the baselines in terms of total score/computation time, using a representative set of mazes (iii) a discussion over your approach and possible improvements.

4 Evaluation

The evaluation of your deliverables is based on the following criteria :

- Clarity of your report
  - Avoid long and vague sentences and be straight to the point. Ideally, the length of each sentence should not exceed a line.

[2]https://github.com/glouppe/info8006-introduction-to-ai/tree/master/pacman/pmagent.py
[3]https://github.com/glouppe/info8006-introduction-to-ai/tree/master/pacman/randomagent.py
[4]https://www.python.org/dev/peps/pep-0008/

  - Do not hesitate to illustrate with examples.
  - Do not overload your graphics, and write short and clear captions. The reader should understand them in a quick look.
  - Lack of structure leads to lack of clarity. We recommend you to design your report outline on top of the one described in the previous section.
- Clarity and structure of the source code
  - Avoid single-file long code source, and prefers to use a multiple-file modular architecture.
  - Name your variables-attributes-classes according to their usage. Comment your code so that explanations are concise and clear enough to allow the reader to understand the semantics in a quick look.
- Performance of your agent
  - Your agent needs to be better than the baseline algorithms, even in a computational budget time.

These criterions are all importants. We particularly value well written reports and code documentation.
⚠Plagiarism is checked and sanctioned by a 0 grade. It is fine to reuse external code, as long as (i) it does not cover your whole approach, (ii) you fully understand the principles behind and (iii) you do give credits in your code.