The University of Melbourne
School of Computing and Information Systems
COMP90054 AI Planning for Autonomy

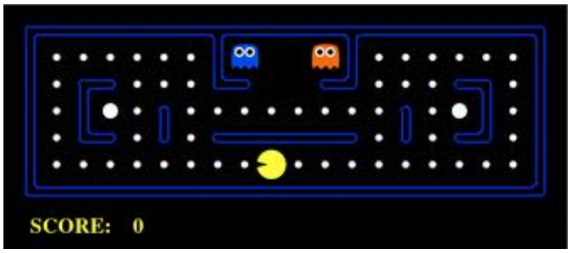
Project 1, 2020

Deadline:
Initial submission: Wednesday 25th March 18:00
Final submission: Wednesday 1st April 18:00

This project counts towards 8% of the marks for this subject.
This project must be done individually.

Aims

The aim of this project is to improve your understanding of various search algorithms using the Berkely Pac Man framework.



https://inst.eecs.berkeley.edu/~cs188/fa18/project1.html

Your task

Your tasks relate to the assignment at https://inst.eecs.berkeley.edu/~cs188/fa18/project1.html.

Create a private repository (0 marks)

Before you start your assignment, you'll need to create a gitlab repository on the MSE servers. Navigate to https://gitlab.eng.unimelb.edu.au and register an account, then use the 'New Repository' button to create a repository for your project work. **Make sure you select a 'Private' repository**. This will ensure your repository can only be accessed by you and by the teaching staff, otherwise your repository may be viewable by others in the class.

Once done, please fill in this form as we will use the details you submit to find your project for marking.

Finally, please fork the following repository into your own git repository as this contains the framework code you need to modify: https://gitlab.eng.unimelb.edu.au/cewin/comp90054-2020-a1.

These tasks must be completed by the initial submssion date.

Practice Task (0 marks)

To familiarise yourself with basic search algorithms and the Pacman environment, it is a good start to implement the tasks at https://inst.eecs.berkeley.edu/~cs188/fa18/project1.html, especially the first four tasks; however, there is no requirement to do so. We have implemented the 'Depth First Search' task for you to help you understand how to interact with the project framework.

Part 1 (3 marks)

Implement the Enforced Hill Climbing algorithm discussed in lectures, using Manhattan Distance as the heuristic. You should be able to test the algorithm using the following command:

```
python pacman.py -l mediumMaze -p SearchAgent -a fn=ehc
```

Other layouts are available in the `layouts` directory, and you can easily create you own!

Part 2 (3 marks)

The Iterative Deepening A* algorithm is a search algorithm that we have not introduced in lectures. It is similar to the Iterative Depth Search algorithm, but but uses the $f(n) = g(n) + h(n)$ value described in A* to control the maximum depth of the search tree at each iteration. This allows it to explore promising nodes more deeply, rather than going to the same depth for each branch of the tree. It is often used to find optimal solutions for memory-constrained problems. You can find out more information on the Iterative Deepening A* algorithm e.g. on Wikipedia. Implement the Iterative Deepening A* Search algorithm. You should be able to test the algorithm using the following command:

```
python pacman.py -l mediumMaze2 -p SearchAgent -a fn=ida
```

For this exercise, you can assume we do not need to check for cycles in the problem. Other layouts are available in the `layouts` directory, and you can easily create you own!

Part 3 (2 marks) - Challenge Question

Note that this is a much more difficult question that requires you to interpret and implement an algorithm from a research paper. Learning to implement it successfully will give you great experience in solving a modern AI planning problem and experience in self-directed learning – something that is valuable in general, but particularly with contemporary AI techniques, but is not necessary in order to do well in the subject. As a result there is only a small mark allocation for this question.

Deceptive path-planning involves finding a path to a goal that makes it difficult for an outside observer to guess what that goal might be. This paper by Masters and Sardina describes a number of algorithms for deceptive path-planning [1]. The main idea is that an agent has a true goal as well as one or more false goals. The agent plans a path to the true goal designed to make it difficult for an observer to figure out whether it is trying to reach the true goal or one of the false goals.

Your task is to implement two of the strategies described in [1] using the Pacman framework. For this exercise, you may assume there is exactly one true goal and exactly one deceptive goal. For each of the deceptive path planning layouts, assume that the true goal is represented by the Food and that the false goal is represented by a Capsule.

a) {1 mark} Implement an agent that uses the π_{d2} strategy from the Masters and Sardina paper. You can call your agent using the following command

```
python pacman.py --layout testDeceptive2 --pacman DeceptiveSearchAgentpid2
```

b) {1 mark} Implement an agent that uses the π_{d3} strategy from the Masters and Sardina paper, using a value of $\alpha = 2$. You can call your agent using the following command

```
python pacman.py --layout testDeceptive2 --pacman DeceptiveSearchAgentpid3
```

NOTE: You should not change any files other than `search.py` and `searchAgents.py`. You should not import any additional libraries into your code. This risks being incompatible with our marking scripts.

Checking your submission

Run the command:

`python autograder.py`

to run sanity checks for our tests, called `comp90054-part1`, `comp90054-part2`, `comp90054-part31` and `comp90054-part-32`. It is important that you are able to run the autograder and have these tests pass, otherwise, our marking scripts will NOT work on your submission.

Marking criteria

This assignment is worth 8% of your overall grade for this subject. Marks are allocated according to the breakdown listed above, based on how many of our tests the algorithms pass. No marks will be given for code formatting, etc.

Originality multiplier

We will be using a code similarity comparison tool to ensure that each student's work is their own. For code that is similar to another submission or code found online, an originality multiplier will be applied to the work. For example, if 20% of the assessment is deemed to have been taken from another source, the final mark will be multiplied by 0.8.

Late submission policy

Submissions that are late will be penalised 1 mark per day, up to a maximum of 5 marks."

Submission

Initial Submission (0 marks)

By the initial submission date, you must have completed the tasks in the 'Creating a private repository' section. On this date we will clone your repository to ensure that we are able to access it.

Final Submission

The master branch on your repository will be cloned at the due date and time.

From this repository, we will copy *only* the files: `search.py` and `searchAgents.py`. Do not change any other file as part of your solution, or it will not run. Breaking these instructions breaks our marking scripts, delays marks being returned, and more importantly, gives us a headache.

Note: Submissions that fail to follow the above will be penalised.

Academic Misconduct

The University misconduct policy[1] applies. Students are encouraged to discuss the assignment topics, but all submitted work must represent the individual's understanding of the topic. The subject staff take academic misconduct seriously. In the past, we have prosecuted several students that have breached the university policy. Often this results in receiving 0 marks for the assessment, and in some cases, has resulted in failure of the subject.

Important: As part of marking, we run all submissions via a code similarity comparison tool. These tools are quite sophisticated and are not easily fooled by attempts to make code look different. In short, if you copy code from classmates or from online sources, you risk facing academic misconduct charges.

But more importantly, the point of this assignment is to have you work through a series of foundational search algorithms. Successfully completing this assignment will make the rest of the subject, including other assessment, much smoother for you. If you cannot work out solutions for this assignment, submitting another person's code will not help in the long run.

References

[1] Masters, P., and Sardina, S. Deceptive path-planning. pp. 4368–4375.

[1] See https://academichonesty.unimelb.edu.au/policy.html