University of Science
Introduction to AI

Lab 1: Search in Graph

`DFS, BFS, UCS, AStar`

Nguyen Bao Long
baolongnguyen.mac@gmail.com

Abstract

The goal of this lab is to teach you about the "framework" of search algorithms in graph. There are some algorithms included: `DFS, BFS, UCS, AStar`.

1 Submission guideline

You will do this assignment on your own. At the beginning, create a folder named `ID` (e.g. If you ID is 123456, then your folder named `123456`) with the following contents:

- A folder named `source` that contains all of your source code.
- A file named `video.txt` that contains the URL of the video that demonstrates the algorithms
- A file named `report.pdf`. You can write your report using Word or anything else. But I suggest using Latex with this template (this is thesis template of University of Science).

Compress the folder (so you can get `123456.zip`) and submit on Moodle.

2 Evaluation

2.1 Criteria

- Study and present search algorithms in graph: 4p
- Compare the algorithms with each other: 2p
- Implement the algorithms: 3p
- Research, present, compare and implement other search algorithms: 1p

2.2 Notes

- Citation is a must. If you refer to a document, you should cite it. Do not cite `ChatGPT`!
- You can discuss with each other, but your work must be yours.
- If you violate one of these notes, you will get a zero point.

3 Requirements

3.1 Studying & Presenting algorithms

- State the search problem. List out the elements of a search problem. Write a general pseudo code to solve a search problem. Distinguish Uninformed Search and Informed Search.
- Study 4 algorithms - `DFS, BFS, UCS, AStar` with the following requirements:
  - General idea

- Pseudo code
- Analyze the algorithm: completeness, optimality, complexity. Regarding to AStar, offer some heuristic functions. What is the admissible heuristic?
- Illustrate the pseudo code by a simple example: A graph with 5-6 nodes. You should use the same graph for algorithms in order to easily compare them

3.2 Comparison

- Compare `UCS, Greedy, AStar`
- Compare `UCS, Dijkstra`

3.3 Implementation

Your mission is to find the path between the orange node and the purple node in the maze. I suggest you play around with the code (such as changing the probability of the brick's appearing, modifying the direction of the agent to 4 directions instead of 8, trying some new heuristic functions, design a dinosaur,...) before handing on to the assignment. Trust me! It's fun!

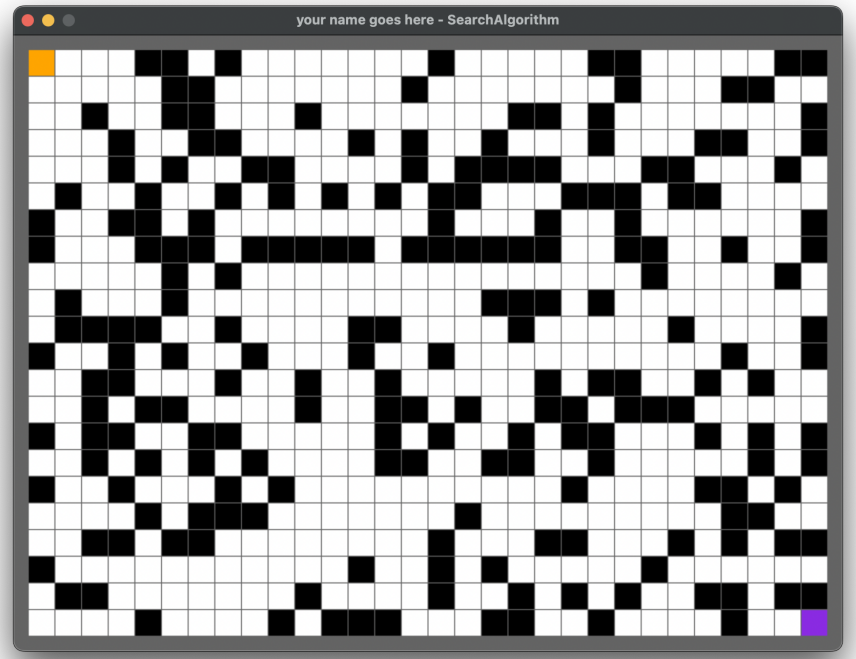

Figure 1: The maze with start node (orange) and goal node (purple). When you work on the code, it might be a different maze, since I change the seed or something else.

There are some **hints and code explanation** that I would like to provide you in advance:

- `const.py`: This file contains all the pre-defined constants. Do not change the colors and the number of nodes. You can change the height and width of the windows in order to fit your laptop

- `maze.py`: This file defines the search space, aka the maze.
- You can write more utility functions in `algos.py` if needed, but do not change the parameters of my functions. Some of the following information might be useful for you:
  - `open_set`: The set which contains the nodes that could be discovered
  - `closed_set`: The set which contains the discovered nodes
  - `father`: `father[x] = y` means that you can go to node `y` from `x`. It would help you on tracing the path when you reach the goal
  - `cost`: `cost[x] = 10` means that the cost to reach `x` from start state is 10
  - You also can design you own `open_set, closed_set, father, cost` without using mine
  - `python main.py --algo AStar` is to run the `AStar` algorithm

For the implementation section, you are going to:

- Implement `DFS, BFS, UCS, AStar`
- For each algorithm, capture the result (screenshot) and describe shortly about the search process. Comment on the result
- Record the running process of all algorithms into a video (5 minutes at most). Please clearly segment the video into each algorithm. Upload that video on YouTube and send me the link in `video.txt`.

Important notes:

- You should create a virtual environment for this assignment using `conda`. There should be **Python 3.10.x** and **Pygame 2.1.x** running on that environment
- When you work on the assignment, do not change my code
- Have fun!