Lab 1: Uninformed Search & Informed Search

Introduction to Artificial Intelligence

Instructor: Nguyen Bao Long

13/03/2023

Abstract

Study, report and then implement searching algorithms in graph.

1 General information

1.1 Grading

- In this assignment, you will learn how to perform a searching algorithm in graph. A general objective of all courses is for you to learn, truly learn. That means you can discuss with your classmate but **the work submitted must be yours**. Therefore, cite all your references and do not cheat or you will be graded 0.
- Specifically, there are 4 parts involving in this assignment:
  - Study and report about searching algorithms in graph (4p).
  - Compare these algorithms to each other (2p).
  - Implement these algorithms (3p).
  - Study about other searching algorithms. Note that your study must meet the above criteria (1p).

1.2 Submission guidelines

- Create a folder named `ID` (e.g. `123456`) that contains:
  - Your report: Named as `ID.pdf` (e.g. `123456.pdf`).
  - Your source code: Put your source code in folder `./src`.
  - A Youtube URL: Specify the link to your video demo in the report. Note that the title of the video should be `[Search in graph] - ID - HCMUS` (e.g. `[Search in graph] - 123456 - HCMUS`).
- Compress that folder → `ID.zip` (e.g. `123456.zip`) and submit on Moodle.

2 Requirements

2.1 Study & Report

- State components of a search problem. Differentiate Informed search and Uninformed search.
- State the solution (in pseudo-code) of a search problem.
- Report about 4 searching algorithms DFS, BFS, UCS, A^*. The following contents should be included:
  - General idea of the algorithm.
  - Pseudo-code.
  - Properties of algorithm (completeness, optimal, time complexity, space complexity).
  - Definition of *heuristic* in A^*. List some options of heuristic.

2.2 Comparison

- Compare between $UCS, Greedy$ and A^*.
- Compare between UCS and $Dijkstra$.

2.3 Implementation

- Your mission is to find the path between 2 nodes in the graph in Figure 1:

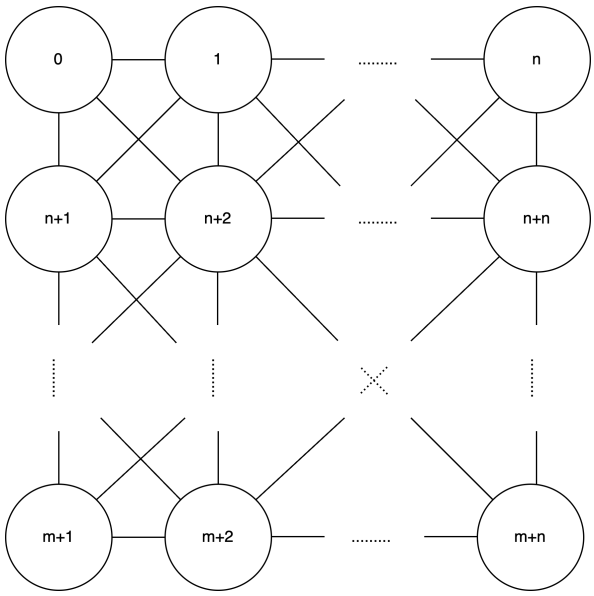

Figure 1: Graph

2.3.1 Source code explanation

- File `./src/Space.py` defines a state space, contains `class Node` and `class Graph`.
  - `class Node` defines a node object and supported functions.
  - `class Graph` defines a graph object and supported functions.
- File `./src/SearchAlgorithms.py` is where you implement searching algorithms. You can design other supported functions as well as use the following hints (or not). However, you must not change the parameters of all given functions. Hints:
  - `open_set`: Contains nodes that are extended to.
  - `closed_set`: Contains nodes that were taken out of `open_set`.
  - `father`: If node `x` extends to node `y`, then `father[y] = x`.
  - `cost`: If cumulative cost from start state to state `x` is `y`, then `cost[x] = y`.
- File `./src/Constants.py`: Contains constants. Do not change the value any variable in this file.
- File `./src/main.py`: You can run the program by calling this file. For example:

```
python main.py --algo AStar --start 71 --goal 318
```

2.3.2 Requirements

- Read the provided source code and complete all the un-implemented function (DFS, BFS, UCS, A^*).
- For each algorithm, take screenshot of the final result and give a brief description about its running process.
- Record **1 video demo** that's **up to 5 minutes** and upload on Youtube. You can refer to this video. Note that you **have to follow the coloring rules for the nodes and the found path** (read file `./src/Constants.py`).

3 References

- Artificial Intelligence: a Modern Approach, EBook, Global Edition.
- CS 188 | Introduction to Artificial Intelligence, Fall 2018: Week 1 - Uninformed Search & A* Search and Heuristics.