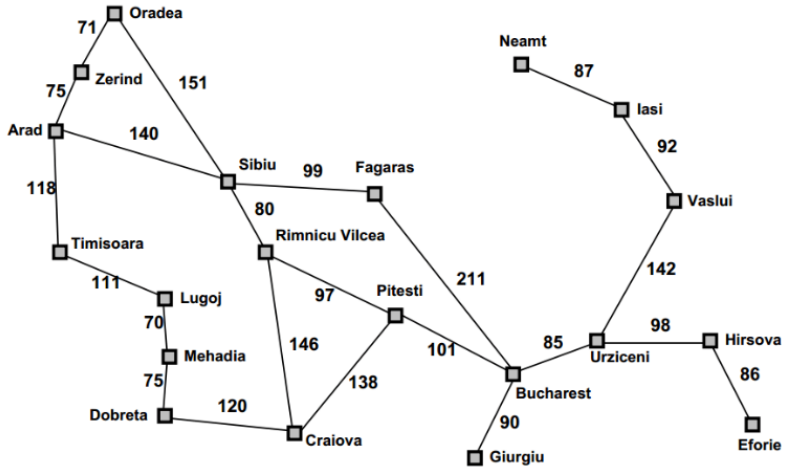
CSCI218 Foundations of Artificial Intelligence

Assessed Lab 2 (5%)

Due date:

Solving Problems by Searching



Overview

Searching is an important approach used by intelligent agents to solve numerous problems including the travelling salesman problem, the 8-puzzle problem, the N-queens problem, and any practical problems that can be formulated as a searching task in a state space. In this subject, we have studied two types of search methods and their specific algorithms, which are listed as follows.

- Uninformed search: Breadth-first search, Depth-first search, and Uniform-cost search.
- Informed search: Best-first search, Greedy best-first search, and A* search.

In this lab, you will apply them to solve the problem of travelling according to a simplified road map of Romania. You are provided with the Python code that has implemented all the search functions and the simplified road map.

What you need to complete this lab:

1. The code (**assessed_lab_2.py** and **utils.py**) taken from the main textbook's website [1].
2. The lecture content and recordings in Week 8.
3. Python 3 programming environment with required libraries, packages, and modules.

Objectives

- Understand the concept of Solving Problems by Searching and its implementation.
- Learn to choose classic searching methods and algorithms.
- Learn to conduct searching on a typical searching task.
- Learn to use libraries, packages and modules related to searching problems.

Questions (5 marks)

The sample code '*assessed_lab_2.py*' includes seven functions for search algorithms (e.g., breadth first tree search, depth first tree search, and A*). Assume that the initial state is '*Fagaras*' and the goal state is '*Zerind*'. Please complete the **main function** to call **any five** of the search functions.

- Run your code and report the results by completing the following table.

ID	Algorithm	Explored states	Solution	Path cost	Time (seconds)
1					
2					
3					
4					
5					

- Your report must clearly describe the five selected search algorithms and provide a **detailed analysis of the results**, including how the LIFO stack and FIFO queue work for each expansion.
- Each correctly implemented search algorithm and its analysis is worth **1 mark**.

Submission

- Submit **a single PDF file** which contains your answers to the questions of all tasks. All questions are to be answered. A clear and complete explanation needs to be provided with each answer.
- The PDF must contain typed text of your answers (**do not submit a scan of a handwritten document**. Any handwritten document will be ignored). The document can include computer generated graphics and illustrations (hand-drawn graphics and illustrations will be ignored).
- The PDF document of your answers should be no more than 5 pages including all graphs and illustrations. Appendix is allowed and will not be counted for the page limit.
- The size limit for this PDF report is 20MB.
- Submit **your Python code** for the lab task separately. Note that, your code must be in Python fomat (i.e., PY file) and well-documented. Other file formats (e.g., PDF and DOCX) will be not evaluated.
- ZIP all the files into **a single zip file and submit it via the submission link on Moodle**.
- **Late submission will not be accepted** without academic consideration being granted.

References

[1] https://github.com/aimacode/aima-python/blob/master/search.py.

**** END ****