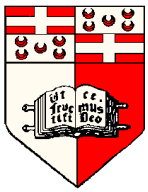

Data Structures and Algorithms 2, Course Project 2017

State-Space Search Algorithms

Creating random graphs:

- Create a graph containing n nodes, where n is a random number between 10 and 12 inclusive.
- For each node, create m distinct directed edges pointing to a random node. Notes:
  - m is a random number between 1 and 4 inclusive.
  - Label each edge with a real-valued weight between 0 and 1 inclusive.
  - Do not create multiple edges between the same ordered pair of nodes (we do not want a pseudograph).
- Designate a random node as the *starting node*.
- Designate 3 distinct random nodes as *goal nodes*.
- Before choosing your graph representation (e.g. an adjacency list, adjacency matrix, …) read what is coming next.

State-space search:

- Implement the following algorithms to search the graph starting from the starting node to one of the goal nodes:
  - Depth-first search, breadth-first search, uniform-cost search, iterative deepening search, A* search, and iterative deepening A*.
- Each algorithm should return and display a path (if it exists) from the starting node to the goal node as well as the cost of the path.
- When implementing your algorithms make sure you deal with cycles in your graph – they can cause lots of problems.

Behaviour of the artifact:

- Each time the program is run, a random graph should be created, and each of the search algorithms must run on it.
- Your program must display the number of nodes, and the number of edges in the graph.
- Your program must display the path found and its cost for *each* algorithm.

Your report:

- Understand the concepts of completeness and admissibility in state space search algorithms.
- Describe each algorithm and comment on its time and space complexity, as well as its completeness and admissibility.
- Make sure to justify the graph representation you choose.

Mark distribution:

- 50% - successful implementation.
    - 5% DFS, 5% BFS, 10% IDS, 10% UCS, 10% A*, and 10% IDA*.
- 50% - report quality and observations on each algorithm.
    - 5% DFS, 5% BFS, 10% IDS, 10% UCS, 10% A*, and 10% IDA*.

Helpful material:

- Book: Artificial Intelligence: A Modern Approach – Russel and Norvig.
- Video lectures (YouTube): Tree search algorithm series – Wheeler Ruml, University of New Hampshire.
- Video lectures (YouTube): Iterative Deepening and Iterative Deepening A* – Alan Mackworth, University of British Columbia.
- Video lectures (YouTube): Depth First, Hill Climbing and Beam Search – Patrick Winston, Massachusetts Institute of Technology.
- The library and the internet.

Other notes:

- Your submission on VLE should include your report in PDF as well as your source code and executable.
- In general, I am not concerned with which programming language you use to implement this project. However, unless you develop your artifact in BASIC, C, C++, Objective C, Swift, Pascal, Java, C#, Matlab, or Python, please consult with me to make sure that I can correct it.
- Provide an executable version of your program (in general, I expect to "double click" on your executable and be able to run your program – assume I'm running a clean installation of Mac OS X, or Windows 10).
- Do not create an installer.
- This is not a group project.
- Plagiarism will not be tolerated.
- Include the plagiarism declaration form in the beginning of your report.
- Any other University of Faculty project requirements and regulations apply.

The deadline for this assignment is Friday 26th May 2017

You will be expected to submit on VLE.