School of Computer Science, University of St Andrews, CS5011, 2021-22

CS5011: A1 - Search - Marine Navigation Planner

Assignment: A1 - Assignment 1

Deadline: 9th of February 2022

Weighting: 25% of module mark

Please note that MMS is the definitive source for deadline and credit details. You are expected to have read and understood all the information in this specification and any accompanying documents at least a week before the deadline. You must contact the lecturer regarding any queries well in advance of the deadline.

---

1 Objective

This practical aims to implement and evaluate a number of AI search algorithms applied to the task of a marine navigation route planner.

2 Competencies

- Design, implement, and document AI search algorithms.
- Compare the performance of AI search algorithms.
- Test methods for optimising AI search.

3 Practical Requirements

3.1 Introduction

A passage plan is a document which indicates a vessel's planned route, including information such as port of departure, arrival port, route, time, tides, weather forecasts etc[1]. A marine navigation route planner is a system that helps compute the best route for a vessel. There are numerous marine navigational apps and simulators to be used for professional or recreational purposes.

In this practical, we focus on a simplified version of a marine navigation route planner for autonomous ferries to navigate the sea surrounding the Republic of Izaland and connecting its 400 islands[2]. An autonomous ferry is used to transport passengers from an island to another without human intervention. To better plan waypoints on the navigation charts, we will use a map inspired by a 2D triangular mesh[3], where a portion of Izaland's sea and land is represented via a 2D triangular grid. Our route finder planner is to be operated by an agent whose aim is to find the best route from the departure port to the destination port, and we refer to this simply as a ferry agent in the spec. The agent moves from the centre of a triangle to the centre of an adjacent triangle but some triangles cannot be crossed as they represent portions of land.

[1] `https://en.wikipedia.org/wiki/Passage_planning`

[2] `https://wiki.opengeofiction.net/index.php/Izaland#Waterways`

[3] `https://en.wikipedia.org/wiki/Triangle_mesh`

3.2 The task

Our agent is equipped with a 2D map where cells are all equilateral triangles of the same size as shown in Figure 1. The agent starts from a cell where we assume there are awaiting passengers at a departure port S, and brings the passengers to a destination port G, in the most efficient way to provide the best possible customer service. Which algorithm should the agent use?

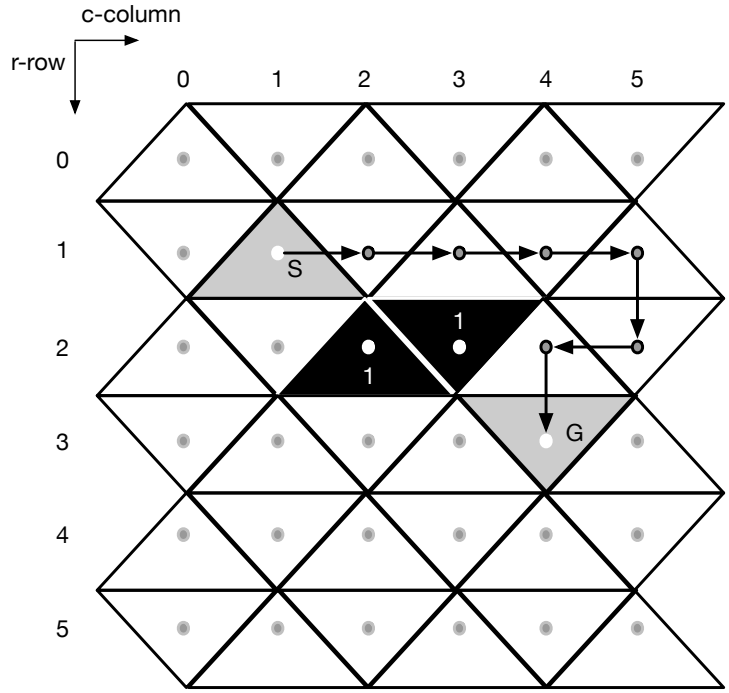

Figure 1: Example map

The ferry agent moves in the navigation map according to the following rules:

i. The map is formed by triangles organised on a square grid of NxN dimensions (where N is a positive integer, N=6 in Fig. 1) where each coordinate in the grid represents the centre of a triangle (marked as grey dots in Fig. 1). The grid is constructed by alternating an upwards pointing triangle with a downwards pointing triangle to fill the N positions in the row. The top row starts with an upwards pointing triangle, and the row continues with alternating triangles shapes. The second row starts with a downwards pointing triangle, etc...

ii. The maps are then simply represented as a two-dimensional coordinate system (r, c) of cells where r is the row and c is the column[4]. Maps for evaluation are provided with the starter code (see Section 4.2) and are represented as Java 2D arrays of integers, `int[][] map=new int[rows][cols]`, where `rows=cols=N`.

iii. Free cells are marked with a '0' (white cells in Fig.1).

iv. The agent starts at cell $S = (r_s, c_s)$ and then terminates at cell $G = (r_g, c_g)$ (e.g., in Fig. 1 S=(1,1), G=(3,4)).

[4]Please note this coordinate system is inverse with respect to a cartesian representation (x, y) where the row number is a y coordinate, and the column number is an x coordinate. The (r, c) representation is in line with the Java representation of 2D arrays.

v. The agent can only move along the grid, from free cell to free cell using the coordinate system. We assume that each movement is from the centre of a cell to the centre of the next cell and that each step has cost 1.

vi. From a cell, the agent moves in the triangle map in 3 directions, Left, Right, and Down if the triangle points upwards and Left, Right, and Up if the triangle points downwards, as shown in Figures 2-3.

vii. Cells marked with '1' in the map represent land, therefore the ferry agent should not cross any of those cells.

viii. The agent cannot go beyond any map boundaries.

ix. If a path cannot be found, the agent must inform the user with a failure message.

Directions:

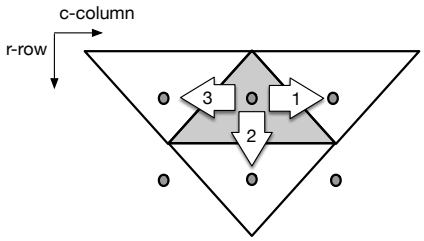

Figure 2: Upwards pointing triangles

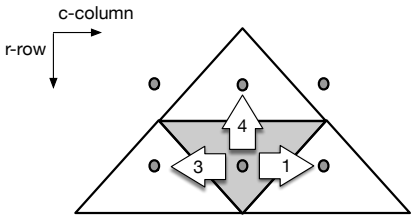

Figure 3: Downwards pointing triangles

The aim of the practical is to implement and evaluate a set of AI search algorithms. There are two main criteria for evaluating search algorithms: the quality of the solution (e.g., the path cost of the agent route), and efficiency (e.g., the number of search states visited).

3.3 Basic Ferry Agent

A basic marine navigation planning agent makes use of uninformed search to identify a route to transport the passengers from a port to another. Implement depth-first search (DFS), and breadth-first search (BFS) following the general algorithm for search. Please ensure that your implementation makes minimal changes between DFS and BFS. The algorithm should also avoid loops and redundant paths.

The tie-breaking strategy to be used for this part is Right, Down, Left, Up, moving clockwise around the triangle. The priority is indicated with a number in the arrows in Figure 2 (1: Right , 2: Down , 3: Left , 4: Up).

For each algorithm, once the search has identified the route to the goal, it must be able to construct the full path.

The output for this part must be the coordinates in the nodes of the frontier at each step before polling a node, within squared brackets separated by a single space, followed by three lines: the path represented as a sequence of (r, c) coordinates with no spaces, the path cost (`double`) and the number of nodes visited (`int`). An example of the output for Figure 1 is formatted as follows:

```
[(1,1)]
[(1,2),(2,1),(1,0)]
.....
(1,1)(1,2)(1,3)(1,4)(1,5)(2,5)(2,4)(3,4)
7.0
24
```

If there is no path, the output should be:

```
[(1,1)]
[(1,2),(2,1),(1,0)]
....
fail
34
```

No other output should be printed.

3.4 Intermediate Ferry Agent

An intermediate marine navigation planning agent extends a basic one by making use of informed search. For many search problems, better results can be obtained by using heuristics to choose the state to explore next. Implement best-first search (BestF) and A*(AStar) search using one of the Manhattan distance heuristics as discussed in the lectures for both algorithms. Your implementation should follow the general algorithm for search. Tie-breaking strategies do not need to be applied in this part. The output for this part must be formatted in the same way as the basic agent with one modification to the frontier, where for each node we must print the coordinates and the f_cost separated by a colon. An example is provided below for BestF, however, please note that the order of the nodes in the frontier and the f_cost might be different depending on your own implementation.

```
[(1,1):5.0]
[(1,2):4.0,(2,1):4.0,(1,0):6.0]
...
(1,1)(1,2)(1,3)(1,4)(1,5)(2,5)(2,4)(3,4)
7.0
9
```

3.5 Advanced Ferry Agent

It is strongly recommended that you ensure you have completed the Requirements of the basic and intermediate agent before attempting any of these requirements.

An advanced marine navigation planning agent extends an intermediate one with at most two additional functionalities. Please note that there is no benefit for implementing more than two. Examples of additional functionalities include:

1. Find out about another search algorithm, such as bidirectional search, implement it and evaluate it in comparison with the other developed approaches.

2. In addition to moving from a cell to the next, the agent might have to perform additional actions, such as planning or waiting for high/low tides. Extend the approaches in the previous parts to include the implementation of new types of actions and evaluate the algorithms under those conditions. Remember to consider well how to account for these actions in the heuristics for informed search.

3. Consider a team of 2 or 3 agents located in adjacent cells. The agents move simultaneously of one position at a time but cannot be on the same coordinates at the same time, and must move such that they are located in adjacent cells at all times. Adapt any of the previous algorithms developed to perform this task. You can decide whether both agents move every turn or one can stay still.

4. Suggest your own advanced requirements: this must be of similar level of difficulty as those suggested above. For a significant advancement, you may apply your search algorithms to an extended search problem, or identify some other search algorithm and propose its implementation and evaluation to solve the marine navigation problem.

4 Code Specification

4.1 Code Submission

The program must be written in Java and your implementation must compile and run without the use of an IDE. Your system should be compatible with the version of Java available on the School Lab Machines (Amazon Corretto 11 – JDK11). Please note, you are not allowed to use libraries that implement search algorithms, but other libraries that are secondary to the objectives of the practical can be used (e.g., Java Util). Your source code should be placed in a directory called **src/** and should include all non-standard external libraries. The code must run and produce outputs as described in the next sections.

Please note that code that does not adhere to these instructions may not be accepted.

4.2 Starter Code

For this practical, four starter classes are provided `A1main.java`, `Conf.java`, `Map.java`, and `Coord.java`.

`Map.java` provides the navigation maps (MAP) to be used for evaluation, 5 in total, defined as Enum Type[5]. This class also includes a few test maps that are used for discussion in class (JMAP). JMAP00 is the map presented in Figure 1. Please do not modify the maps in this class but you can add more if you wish.

`Coord.java` is a data structure for storing the coordinates, as row and column.

`Conf.java` provides the configurations (CONF) to be used for a specific run, 25 in total, also defined as Enum Type. Each configuration includes a map corresponding to the above, and the coordinates of the cells S, and G. Additional test configurations are also included (JCONF). For example, to run the configuration of Figure 1, we use JCONF00. Please do not modify the configurations in this class but you can add more if you wish.

`A1main.java` contains only example code on how to retrieve a map to be used, how to print it for debugging purposes, and some examples of the required output. This class can be modified as required, however, please ensure that the final format of input and output is as that produced in the sample code.

4.3 Running

Your code should run using the following command:

```
java A1main <DFS|BFS|AStar|BestF|...> <ConfID> [<any other param>]
```

For example for BFS in Figure 1 we will use

```
java A1main BFS JCONF00
```

For the basic and intermediate agents, the output of the system must be as described as we will perform some automatic tests on the required output.

[4] `https://aws.amazon.com/corretto/`

[5] `https://docs.oracle.com/javase/tutorial/java/javaOO/enum.html`

4.4 Automatic Testing

We provide some very basic unit tests to help you check that your output is structured with the required format. The tests only check the BFS strategy. They can be found at `/cs/studres/CS5011/Practicals/A1/Tests` and can be run with `stacscheck`.

In order to run the automated checker on your program, place your program in a directory in the School lab machines, ssh into one of the School computers running Linux if you are working remotely, then change directory to your CS5011 A1 directory and execute the following command:

```
stacscheck /cs/studres/CS5011/Practicals/A1/Tests
```

Please note that these tests are not sufficient to check that your program works correctly, please do test your system well and provide information on your own testing in the report.

5 Report

You are required to submit a report describing your submission in PDF with the structure and requirements presented in the additional document *CS5011_A_Reports* found on `studres`. The report includes 5 sections (Introduction, Design & Implementation, Testing, Evaluation, Bibliography) and has an advisory limit of 2000 words in total. The report should include clear instructions on how to run your code. Consider the following points in your report:

1. Describe the search problem components for the marine navigation planning problem.
2. Describe the implementation of the search strategy clearly, and the differences between the various algorithms in your implementation.
3. Include details on how the successor function works.
4. When evaluating the algorithms, which algorithm is best in terms of the number of search states visited? Which algorithm produces the best routes on the basis of path cost?

6 Deliverables

A single ZIP file must be submitted electronically via MMS by the deadline. Submissions in any other format will be rejected.

Your ZIP file should contain:

1. A PDF report as discussed in Section 5
2. Your code as discussed in Section 4

7 Assessment Criteria

Marking will follow the guidelines given in the school student handbook. The following issues will be considered:

- Achieved requirements
- Quality of the solution provided
- Examples and testing

- Insights and analysis demonstrated in the report

Some guideline descriptors for this assignment are given below:

- For a mark of 8 or higher: the submission implements part of the basic agent able to find a path in at least one way, adequately documented or reported.
- For a mark of 11 or higher: the submission implements the basic agent fully. The code submitted is of an acceptable standard, and the report describes clearly what was done, with good style.
- For a mark of 14 or higher: the submission implements fully the basic agent with a good attempt to the intermediate agent. It contains clear and well-structured code and a clear report showing a good level of understanding of design and evaluation of search algorithms.
- For a mark of 17: the submission implements fully the basic and intermediate agent. It contains clear, well-designed code, together with a clear, insightful and well-written report, showing in-depth understanding of design and evaluation of search algorithms.
- Above 17: the submission implements fully the basic and intermediate agent and one or two advanced agent functionalities. The submission should demonstrate unusual clarity of implementation, together with an excellent and well-written report showing evidence of extensive understanding of design and evaluation of search algorithms.

8 Policies and Guidelines

Marking: See the standard mark descriptors in the School Student Handbook

`https://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/feedback.html#Mark_-Descriptors`

Lateness Penalty: The standard penalty for late submission applies (Scheme B: 1 mark per 8 hour period, or part thereof):

`https://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/assessment.html#latenesspenalties`

Good Academic Practice: The University policy on Good Academic Practice applies:

`https://www.st-andrews.ac.uk/students/rules/academicpractice/`

Alice Toniolo, Nguyen Dang, & Mun See Chang

cs5011.lec@cs.st-andrews.ac.uk

January 20, 2022