EAI 320

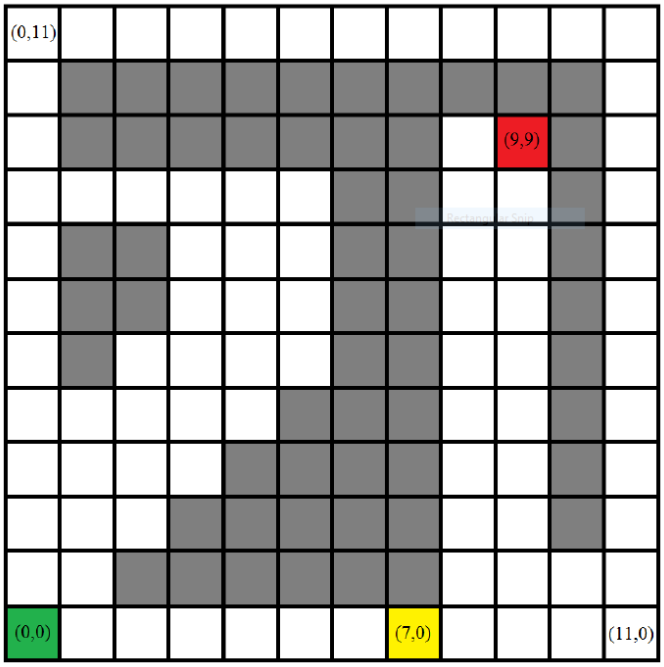
Practical Assignment 2

Concept by Hugo Coppejans
Compiled, edited, and reviewed by Johan Langenhoven

15 Februrary 2018

1 Scenario

You have been assigned to design an AI agent for a computer game. The goal of the agent is to find a path from a starting point to a specified goal point. The environment is given as follows:



The green block is the starting point. The red block is the goal point. The grey blocks are walls. The agent cannot navigate to or over these blocks. The yellow block is used for question 2. Ignore it in question 1.

The agent can move vertically, horizontally and diagonally, one block at a time. This implies that, with exception to locations adjacent to the environment edges and the walls, the agent can move in 8 possible directions.

In python, this environment could be represented by the following list:

```
[[0,0,0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,1,1,1,0,1,1,0],
 [0,1,0,0,0,0,1,1,0,1,1,0],
 [0,1,1,0,0,0,0,0,0,1,1,0],
 [0,1,1,1,0,0,0,0,0,1,1,0],
 [0,1,1,1,1,0,0,0,0,1,1,0],
 [0,1,1,1,1,1,1,1,1,1,1,0],
 [1,1,1,1,1,1,1,1,1,1,1,0],
 [0,0,0,0,0,0,0,0,0,0,1,0],
 [0,0,0,0,0,0,0,0,0,0,1,0],
 [0,0,1,1,1,1,1,1,1,1,1,0],
 [0,0,0,0,0,0,0,0,0,0,0,0]]
```

Note: the list might seem reversed, but take a careful look at where [0][0] is on the map.

This is a tree search problem. The root node describes the state where the agent is located at (0,0). The tree is expanded by moving the agent. The next layer in the tree thus consists of the locations (0,1), (1,1) and (1,0). Each of these nodes can then be expanded. (Don't try grow the whole tree). Read through all the questions before you start.

Task 1

Question 1a

Implement the **uniform cost** search to find the path from (0,0) to (9,9). Use the Euclidean distance to assign a cost for moving from a current square to its neighbour. This will result in a cost of 1 to move vertically or horizontally and a cost of sqrt(2) to move diagonally. The total path cost at a specific node is an accumulation of the cost of each move as you traverse down the tree. The total path cost is labelled as $g(n)$.

Question 1b

Implement the **greedy/best first** search to find the path from (0,0) to (9,9) using the Euclidean distance from the current position to (9,9) as a heuristic. The heuristic is labelled as $h(n)$.

Question 1c

Implement the **A*** search to find the path from (0,0) to (9,9). Use the heuristic used in question 1b and the path cost used in question 1a. A* uses the cost of $f(n) = g(n)+h(n)$ to determine which node to expand.

Question 1d

Document the results. In your results, for each method include a plot consisting of:

1. The path that is found,
2. The nodes that were explored (or expanded), and
3. The walls.

It is up to you to decide how you want to present the plots. An example of a single image containing everything, made using simple print statements in python, and the highlighter from the "Snippet" application found in Windows, can be seen below. "S" is the Starting node, and "G" is the Goal node.

An example map, and the way to present it.

- | indicate walls.
- 0s indicate unvisited nodes.
- 1s indicate viewed nodes.
- 2s indicate expanded nodes.
- *s indicate the best path.

Question 1e

Provide a discussion which compares the methods in terms of which finds the shortest path and why.

Task 2

Assume that the yellow block at (7,0) is a wall. Rerun your algorithms and compare the results of the uniform cost search, the greedy/best first search algorithm and the A* algorithm for this case, as is done in Questions 1a - e.

Task 3

Discuss the properties of the implemented algorithms as follows:

1. **Completeness:** Do the algorithms find the solution? Under what circumstance would the algorithms fail to find a solution?
2. **Time:** In your code, count the number of branches in the search tree. Each time a node is expanded, you take a set of actions to form branches to new nodes. You need to count how many times you take an action in total.
3. **Space:** In your code, note how large the open set (or frontier queue see hint 4) grows in each algorithm. Explain how this could become a problem in worlds with larger state spaces.
4. **Optimality:** Discuss under which circumstances each algorithm is optimal.

For simplicity, a table containing some details such as Nodes Viewed, Nodes Expanded, etc, makes for easier marking and conciseness.

Tips and Tricks

1. When expanding a node, you consider the actions you can take. You can move to one of the 8 surrounding blocks if you are not located at the environment edge or a wall. You could write a function that returns the set of neighbours for a given location.

2. When you consider expanding a node, check if its value in the environment matrix is 1. If it is, ignore the node.

3. You need to have a memory structure consisting of the nodes that need to be investigated (open set) and the nodes that have been investigated (closed set). Additionally, you need to keep track of node parents for the path reconstruction when you find a solution. See `https://en.wikipedia.org/wiki/A*_search_algorithm` for an example of pseudo code.

4. The three algorithms are identical other than the fact that they use different cost functions. (Uniform cost search uses $g(n)$, greedy/best first search uses $h(n)$, and A* uses $g(n)+h(n)$). As such, try to construct your tree and algorithms as general as possible to avoid reimplementation.

Report

You have to write a short technical report for this assignment. Your report must be written in LaTeX. In the report you will give your results as well as provide a discussion on the results. Make sure to follow the guidelines as set out in the separate questions to form a complete report.

Reports will only be handed in as digital copies, but a hard copy plagiarism statement needs to be handed in at the following week's practical session (on the final day of the practical submission).

Deliverables

- Write a technical report on your finding for this assignment.
- Include your code in the digital submission as an appendix.

Instructions

- All reports must be in PDF format and be named report.pdf.
- Place the software in a folder called SOFTWARE and the report in a folder called REPORT.
- Add the folders to a zip-archive and name it `EAI320_prac2_studnr.zip`.
- All reports and simulation software must be e-mailed to **up.eai320@gmail.com** no later than 16:00 on 22 February 2018. No late submissions will be accepted.
- Use the following format for the subject header for the email: EAI 320 Prac 2 - studnr.
- Upload your report *without* the plagiarism statement to TurnItIn before 16:00, 22 February 2018.
- Bring your plagiarism statement to the practical session on Thursday, 22 February 2018, where they will be collected.
- No hard-copy of the report is required.

Additional Instructions

- Do not copy! The copier and the copyee (of software and/or documentation) will receive zero for both the software and the documentation.
- For any questions or appointments email me at **up.eai320@gmail.com**.
- Make sure that you discuss the results that are obtained. This is a large part of writing a technical report.

Marking

Your report will be marked as follow:

- 60% will be awarded for the full implementation of the practical and the subsequent results in the report. For partially completed practicals, marks will be awarded as seen fit by the marker. **Commented code allows for easier marking!**
- 40% will be awarded for the overall report. This includes everything from the report structure, grammar and discussion of results. The discussion will be the bulk of the marks awarded.