BBM405 – Fundamentals of Artificial Intelligence - Spring 2021

Homework 1

Due date: April 30, 2021

Goal:

In this homework, you will analyze different search algorithms to find a path in a maze.

Introduction:

Finding a path to exit from a maze is a fun application of search strategies that we discussed. In this homework, you will first generate your own mazes. Then, on a maze you will use three search strategies to be able to find a path to the exit: Iterative Deepening Search, Uniform Cost Search, and A* search. Finally, you will compare the search strategies based on the number of nodes expanded, time spent to find the path and length of the path. Overall, you will be able to analyze search strategies from different perspectives similar to the examples shown in https://emmilco.github.io/path_finder/.

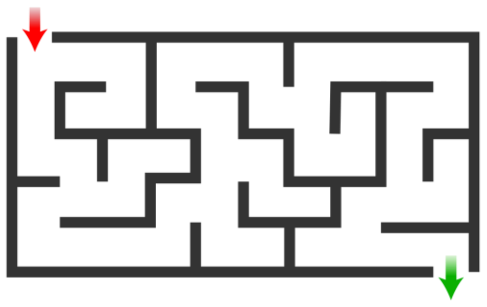
For implementing search algorithms, you can use any language. You can also make use of available codes. Some examples are shown below. You have to be very careful in providing all the references for the resources that you have used.

http://aima.cs.berkeley.edu/python/search.html
https://github.com/chitholian/AI-Search-Algorithms
https://artint.info/AIPython/

You will write a detailed report, and grading of your homework will be based on the information provided in the report. You should also submit all of your codes, mazes, and other resources.

Representation of mazes:

For all the mazes assume that, top left corner is the starting point and bottom right point is the exit. You can only move horizontally or vertically. The edges are assumed to be on the border of the cells. An edge is either a barrier or a blank, that is you cannot move from cell i,j to cell i+1,j if there is barrier in between.



Part 1 (30 pts): Generate your own maze using Randomized DFS

In this part, first you will generate your own maze. To generate a maze, you will apply a randomized Depth First Search (DFS) to the grid as explained in https://www.baeldung.com/cs/maze-generation.

The algorithm is described as below:

"The algorithm starts at a given cell and marks it as visited. It selects a random neighboring cell that hasn't been visited yet and makes that one the current cell, marks it as visited, and so on.

If the current cell doesn't have a neighbor that hasn't been visited yet, we move back to the last cell with a not-visited neighbor.

The algorithm finished when there is no not-visited cell anymore."

The outline of the algorithm is shown below:

Algorithm 1: Randomzied Depth-First-Search

```
1  function createMaze()
2      startVertex ← Vertex(0, 0)
3      randomizedDFS(startVertex)
4  end
5  function randomizedDFS(vertex)
6      markVisited(vertex)
7      nextVertex ← randomUnvisitedNeighbour(vertex)
8      while nextVertex != null do
9          connectCells(vertex, nextVertex);
10         randomizedDFS(nextVertex)
11         nextVertex ← randomUnvisitedNeighbour(vertex)
12     end
13     return
14 end
```

Since the algorithm is recursive it may cause memory issues for big mazes. See https://en.wikipedia.org/wiki/Maze_generation_algorithm for the iterative algorithm

Requirements for Part 1: You are required to generate mazes in five different sizes.

a) 10 x 10
b) 100 x 100
c) 1000 x 1000
d) M x M where M is the last 2 digits of your student ID
e) N x N where N is the last 3 digits of your student ID

Generate 2 different mazes for each of the sizes given above. Show the mazes only for (a) and (d) in your report.

Part 2 (40 pts): Application of search strategies

In this part, you will apply three different search algorithms on mazes.

i) Iterative Deepening Search
ii) Uniform Cost Search
iii) A* search

For A* use the following heuristics

i) Euclidean distance
ii) Manhattan Distance

You will use the mazes that you generated in the first part. You should have 2 randomly generated mazes for each, all together for 5 scales it would be 10 mazes. If you are not able to complete the first part, you can generate mazes in the specified sizes using any generator by providing the reference. You should again provide two mazes for each scale.

For each of the four methods above, run the search algorithm on the mazes. There will be 40 different solutions.

Requirements for Part 2:

In this part of your report, show the solutions as a path from start to the exit only for mazes (a) and (d) for all of the 4 strategies.

Part 3 (20 pts): Analysis of the search strategies

In this part, you will analyze and compare your solutions based on the following:

i) What is the length of the path found
ii) How many nodes are expanded by the algorithm
iii) What is the maximum time taken to find the path

Requirements for Part 3:

In this part of your repot, you should provide a table to compare all 40 solutions based on the 3 criteria specified above.

Part 4 (10 pts): Extending the limits

Finally, based on the capacity of your resources try to extend the sizes of the mazes as much as possible. Compare your results with the complexities discussed in the class. Provide, a detailed analysis based on your observations.

Submission

You must submit a report in pdf format including all the requirements described above. Name the file as BBM405_HW1_Name_Surname.pdf where Name and Surname is yours ☺

You should also submit the following files
-Your mazes generated
-Your solutions for each of the algorithms
-Your codes for Part 1 and Part 2
Send these files in a zip file named as BBM405_HW1_Name_Surname.zip

Good luck