**CAP 6635 Artificial Intelligence**

**Homework 2 [12 pts, Due Feb 12 2023]**

[Homework solutions must be submitted through Canvas. Only pdf, word, and txt files are allowed. If you have multiple pictures, please include all pictures in one Word/pdf file. You can always update your submissions before due date, but only the latest version will be graded.]

1. **[1 pt]** The goal of the 4-queens problem is to place 4 queens on a 4 rows 4 columns chessboard, such that no two queens attach each other. One solution to the 4-queens problem is shown in Figure 1. Design a search-based solution to solve the 4-queens problem. Your solution must include following four components:
   1. Define state and show examples of three states (including initial state and goal state) [0.25 pt]

A state is a specific iteration that has all the information to predict the next state given a specific action. It can also determine if the current state is a goal state. The states in this scenario are the various versions of the chess board every time a Queen is added to the board. The initial state is a blank board, and the goal state is a successful completion of the board that satisfies all rules. All the possibilities as Queens are placed on the board between the initial state and goal state are the other states. I included a diagram under question (c.) I made that displays every state for this problem.

* 1. Define a successor function, and estimate total number of states, based on the defined successor function [0.25 pt]

The successor function will determine the number of states and which states are the successors of a given state. It represents all the actions that are possible in each state. My successor function will weed out future Queen placements that violate the rules of being in the same column, row or diagonal of a previously placed Queen. I estimate that there could be as many as 257 states because there are 4 Queens that need to be placed in 4 rows and in each row there are 4 squares that a Queen can be placed so 4^4 = 256 + 1 state for the initial state. I realize that many of these states will not be realized because they will break the rules and that branch will not continue.

* 1. Show state graph with at least five states and their relationships (the graph must have a path from initial state to goal state) [0.25 pt]

A picture containing text, white

Description automatically generated

* 1. Show how to find solution(s) from the state graph [0.25 pt]

Please see the code I wrote in Python to solve this problem. It will solve for any value of ‘n’ Queens. It will crash my system around 12 Queens problem. It also shows the different states and gives the goal states at the end followed by the number of solutions at the bottom.



Figure 1

A picture containing text

Description automatically generated

Table

Description automatically generated

Shape, circle

Description automatically generated

Figure 2

1. [**2 pts**]. Figure 2 shows a state graph with five states. Assume that “a” is the initial state, and “G” is the goal state.
   1. Use Breath First Search (BFS) to build a search tree to carry out search from the initial state to the goal state (Show complete search tree). [1 pt]
      1. Mark order of the search node (on the BFS tree) being visited from initiate state to find the goal state (break tie in favor of node with a lower alphabetic order, e.g., “b” has a higher priority than “c”).

Text

Description automatically generated

* + 1. Is the solution optimal? Why or why not?

Yes, the solution is optimal because it found the shortest path to the goal state ‘G’. Which was depth = 1.

* + 1. Explain why avoiding revisited state is important in creating BFS search tree.

If you didn’t avoid visited states, then the search would continue searching the same states repeatedly for optimal paths to the Goal even after it had been found and would never terminate.

* 1. Use Depth First Search (DFS) to build a search tree to carry out search from the initial state to the goal state (Show complete search tree). [1 pt]
     1. Mark order of the search node (on the DFS tree) being visited from initiate state to find the goal state (break tie in favor of node with a lower alphabetic order, e.g., “b” has a higher priority than “c”).

A picture containing text

Description automatically generated

* + 1. Is the solution optimal? Why or why not?

Yes, the solution is optimal because it returns the shortest path to the Goal. It This is because the only path to the Goal State is from ‘a’ and when the algorithm encountered a’s in the tree, it ignored them because it already been visited since a was the initial state. Had there been other routes to the Goal state, than this would not have been optimal.

* + 1. Explain why avoiding revisited state is important in creating DFS search tree.

It is important to avoid revisiting states because you could end in an infinite loop of search. For instance, if a had not been marked as visited there would have been an infinite search of a-b-a-b-a-b-a-b-a in this graph.

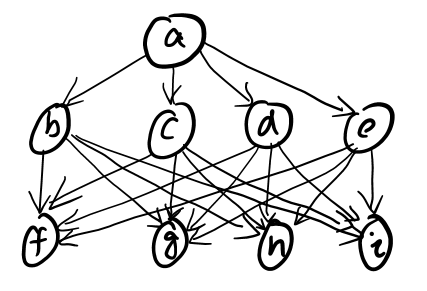


Figure 3

1. **[1.5 pts]** Figure 3 shows a stage graph with 9 states. Create a search tree (using Breadth First Search (BFS) algorithm) to traverse the state graph from “a”.
   1. Show complete BFS search tree with at least three levels (root, 1st, and 2nd). If multiple nodes are created for same state, say “f”, use f1, f2, f3, etc., to denote each node (the number corresponding to node being created). Break tie in favor of node with a lower alphabetic order, e.g., “b” has a higher priority than “c”. [0.5 pt]

Diagram

Description automatically generated with low confidence

* 1. Report the order of the nodes being visited, and the Fringe structure (report fringe structure of each step as show in the table below) [0.5 pt].

Nodes numbered in image

* 1. Compare the fringe structure and explain the memory consumption of each method, with respect to branching factor (*b*) and the search depth (*d*) [0.5 pt].

|  |  |  |  |
| --- | --- | --- | --- |
| Fringe | Node visited | Fringe Size | Search depth |
| A |  | 1 | 0 |
| BCDE | A | 4 | 1 |
| C,D,E,F,G,H,I | B | 7 | 2 |
| D,E,F,G,H,I,F1,G1,H1,I1 | C | 10 | 2 |
| E,F,G,H,I,F1,G1,H1,I1,F2,G2,H2,I2 | D | 13 | 2 |
| F,G,H,I,F1,G1,H1,I1,F2,G2,H2,I2,F3,G3,H3,I3 | E | 16 | 2 |

Branching factor = 4

At d (depth) = 0, the space complexity is b^d = 4^0 = O(1)

At d (depth) = 1, the space complexity is b^d = 4^1 = O(4)

At d (depth) = 2, the space complexity is b^d = 4^2 = O(16)

When these numbers are compared with the fringe structure (specifically its size), they hold true.

The fringe size would be growing by 4 each time a new node is visited because of the branching factor, but the fringe is also being reduced by 1 when the new node is removed. So the fringe is increasing by b-1 every time a new node is expanded. The final entry on the table is the last node to be expanded at search depth 2. You can see that the fringe has grown to the maximum space complexity of b^d = 16 here.

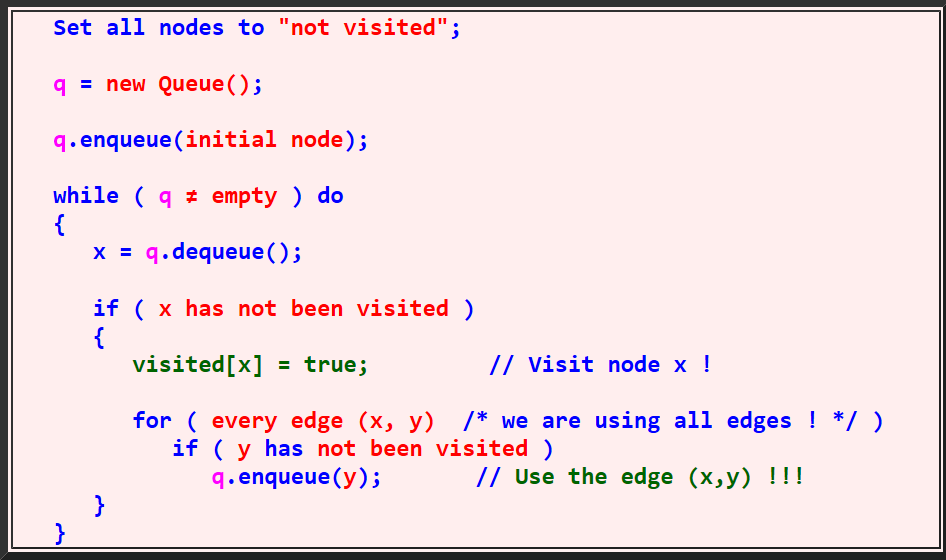


Figure 4: Breath First Search (BFS)

1. **[1.5 pts]** Figure 3 shows a stage graph with 9 states. Create a search tree (using Depth First Search (DFS) algorithm showing in Figure 5) to traverse the state graph from “a”.
   1. Show complete DFS search tree with at least three levels (root, 1st, and 2nd). If multiple nodes are created for same state, say “f”, use f1, f2, f3, etc., to denote each node (the number corresponding to node being created). Break tie in favor of node with a lower alphabetic order, e.g., “b” has a higher priority than “c”. [0.5 pt]

A picture containing text

Description automatically generated

* 1. Report the order of the nodes being visited, and the Fringe structure (report fringe structure of each step as show in the table below) [0.5 pt].

Nodes numbered in image

* 1. Compare the fringe structure and explain the memory consumption of each method, with respect to branching factor (*b*) and the search depth (*d*) [0.5 pt].

The fringe size approaches its upper bound of 8 a few times. This means that it has a high space complexity.

b (branching factor) = 4, d = m (maximum depth) = 2

Space Complexity of DFS = O(b\*m) = 4\*2=O(8)

Text, letter

Description automatically generated

Figure 5 Depth First Search DFS

|  |  |  |  |
| --- | --- | --- | --- |
| Fringe | Node Visited | Fringe Size | Search Depth |
| A | A | 1 | 0 |
| E,D,C,B | B | 4 | 1 |
| E,D,C,I,H,G,F | F | 7 | 2 |
| E,D,C,I,H,G | G | 6 | 2 |
| E,D,C,I,H | H | 5 | 2 |
| E,D,C,I | I | 4 | 2 |
| E,D,C | C | 3 | 1 |
| E,D,I1,H1,G1,F1 | F1 | 6 | 2 |
| E,D,I1,H1,G1 | G1 | 5 | 2 |
| E,D,I1,H1 | H1 | 4 | 2 |
| E,D,I1 | I1 | 3 | 2 |
| E,D | D | 2 | 1 |
| E,I2,H2,G2,F2 | F2 | 5 | 2 |
| E,I2,H2,G2 | G2 | 4 | 2 |
| E,I2,H2 | H2 | 3 | 2 |
| E,I2 | I2 | 2 | 2 |
| E | E | 1 | 1 |
| I3,H3,G3,F3 | F3 | 4 | 2 |
| I3,H3,G3 | G3 | 3 | 2 |
| I3,H3 | H3 | 2 | 2 |
| I3 | I3 | 1 | 2 |

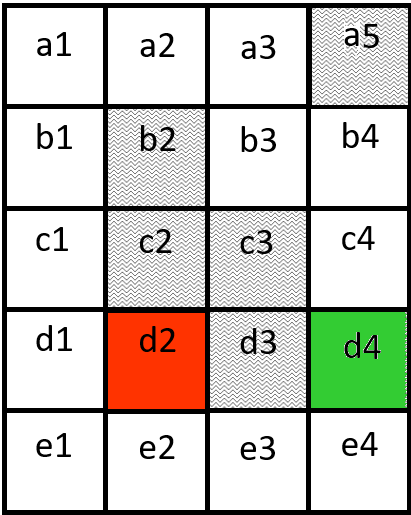


Figure 6

1. [**3 pts**] Figure 6 shows a robot navigation field, where the red square (d2) is the robot, and green square (d4) is the goal. The shaded squares (such as b2, c2, etc.) are obstacles. The robot is not allowed to move in diagonal line. Node are coded using an alphabet letter followed by a digit (such as a0, b1, b2 etc.). When two sibling nodes are inserted into fringe (queue), use deque order to favor node with a lower alphabet and a lower digit. For example, if d1 and e2 are sibling nodes, d1 will be dequeued first (because “d” has a lower alphabetic order than “e”). If a1 and a2 are sibling nodes, a1 will be dequeued first (because “1” has a lower digit than “2”). Node expanded/visited does not need to be revisited.

* Use Depth First Search to find path from d2 to d4.
  + Report nodes in the fringe in the orders they are included in the fringe. [0.5 pt]

\*\*See Chart

* + Report the order of the nodes being expanded. [0.5 pt]

D2, D1, C1, B1, A1, A2, A3, B3, B4, C4, D4, E1, E2, E3, E4

This would either stop at the first D4 or continue if the fringe needs to be empty before termination. Unclear if we should find ALL paths or not.

* + Report the final path from d2 to d4. [0.5 pt]

There are two paths:

1. D2, D1, C1, B1, A1, A2, A3, B3, B4, C4, D4

This is the first path that is found. I wasn’t sure if I was supposed to stop here or keep expanding from the fringe to find potential additional paths. If the goal was to break the search when the goal was reached, this would be the path we would have.

1. D2, D1, E1, E2, E3, E4, D4

This is the second path that is found, but I wasn’t sure if I could go back and keep dequeuing the fringe. This would be the most optimal path if I was, but it was not the first path found.

**DFS**

|  |  |
| --- | --- |
| **Fringe:** | **Node visited/expanded** |
| D2 | D2 |
| D1, E2 | D1 |
| C1, E1, E2 | C1 |
| B1, E1, E2 | B1 |
| A1, E1, E2 | A1 |
| A2, E1, E2 | A2 |
| A3, E1, E2 | A3 |
| B3, E1, E2 | B3 |
| B4, E1, E2 | B4 |
| C4, E1, E2 | C4 |
| D4, E1, E2 | D4 |
| E1, E2 | E1 |
| E2 | E2 |
| E3 | E3 |
| E4 | E4 |
| D4 | D4 |

* Use Breadth First Search to find path from d2 to d4.
  + Report nodes in the fringe in the orders they are included in the fringe. [0.5 pt]

\*\* See Chart

* + Report the order of the nodes being expanded. [0.5 pt]

D2, D1, E2, C1, E1, E3, B1, E4, A1, D4, A2, A3, B3, B4, C4, D4

This would either stop at the first D4 or continue if the fringe needs to be empty before termination. Unclear if we should find ALL paths or not.

* + Report the final path from d2 to d4. [0.5 pt]

There are two paths:

1. D2, E2, E3, E4, D4

This is the first path that is found. I wasn’t sure if I was supposed to stop here or keep expanding from the fringe to find potential additional paths. If the goal was to break the search when the goal was reached, this would be the path we would have.

1. D2, D1, C1, B1, A1, A2, A3, B3, B4, C4, D4

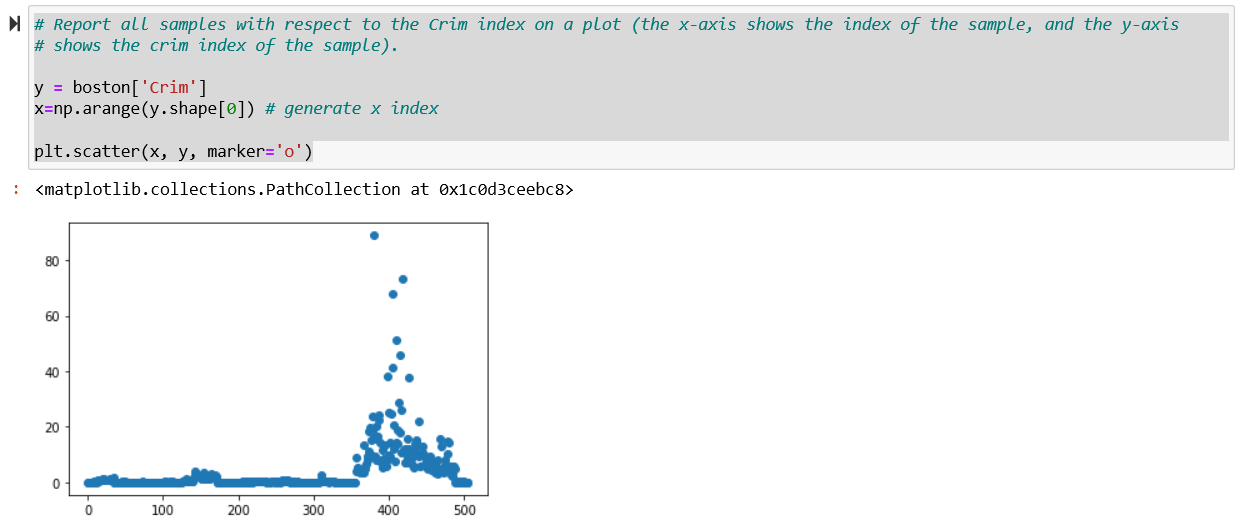
This is the second path that is found, but I wasn’t sure if I could go back and keep dequeuing the fringe. This would be the most optimal path if I was, but it was not the first path found.

**BFS**

|  |  |
| --- | --- |
| **Fringe:** | **Node visited/expanded** |
| D2 | D2 |
| D1, E2 | D1 |
| E2, C1, E1 | E2 |
| C1, E1, E3 | C1 |
| E1, E3, B1 | E1 |
| E3, B1 | E3 |
| B1, E4 | B1 |
| E4, A1 | E4 |
| A1, D4 | A1 |
| D4, A2 | D4 |
| A2 | A2 |
| A3 | A3 |
| B3 | B3 |
| B4 | B4 |
| C4 | C4 |
| D4 | D4 |

**For all programming tasks, please submit the Notebook as html or pdf files for grading (your submission must include scrips/code and the results of the script).**

For each subtask, please use task description (requirement) as comments, and report your coding and results in following format:



1. [**2 pts**] The “**Blind Search to Play Maze [Notebook, html]**” posted on Canvas “Source Code” shows a Maze game using DFS and BFS search (using “DFS” or “BFS” parameters). Use Notebook as the skeleton code, validate and compare following settings and results.
   1. Using Figure 7 as the game field, and use any two states as initial and goal states, report a screenshot of the program. Explaining the meaning of the output [0.25 pt]
   2. Set initial state as [0, 0] and goal state as [0, 1]. Report number of nodes expanded by BFS and DFS, respectively. Report final path discovered by each algorithm, respectively [0.25 pt]. Explain why or why not the method is optimal [0.25 pt], and why one approach expands far more nodes than the other method [0.25 pt]
   3. Set initial state as [0, 0] and goal state as [0, 1], [0, 2], [0, 3], [0, 5], [0, 6], [0, 7], [0, 8], [0, 9], respectively. Create one plot to show the number of maximum Fringe size of BFS (*x*-axis denotes the goal state, and *y*-axis denotes the maximum Fringe size) [0.25 pt]. Create one plot to show the number of maximum Fringe size of DFS (*x*-axis denotes the goal state, and *y*-axis denotes the maximum Fringe size) [0.25 pt]. Explain how the Fringe size grows with respect to the search depth for DFS and BFS, respectively. [0.25 pt]
   4. For goal state [0, 9], compare path returned by BFS and DFS, respectively. Which method is optimal, and which method is not optimal? Why? [0.25 pt]

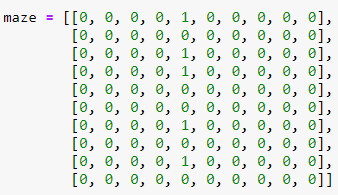


Figure 7: Maze game field

Text

Description automatically generatedText

Description automatically generatedText

Description automatically generated

Text

Description automatically generatedText

Description automatically generatedText

Description automatically generatedText

Description automatically generatedText

Description automatically generatedGraphical user interface, text

Description automatically generatedChart, line chart

Description automatically generatedText

Description automatically generated

1. [1 **pt**] Following Homework 1 ChatGPT assignment, this assignment is to implement a program (ie., using programming) to read questions from an input excel file (e.g., input.csf), and then save the ChatGPT generate answers as another excel file (e.g., output.csv). Each row of the input excel file includes questions to ask ChatGPT (each row corresponds to one question). Your answers should also be saved as a pure text (or excel file), with each row corresponding to one answer. Here are required implementation:
   1. The program should read text from a given excel file (e.g., a csv file), and print out content of the first five lines (using example.csv file posted on canvas as a template) [0.25 pt]

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

* 1. For text content in each row, please use the text as chatGPT input, and collect (and print out) ChatGPT answers [0.5 pt]

Text

Description automatically generated

* 1. Save chatGPT answers as another output.csv file (without each answer being saved as one row). All answers from your input.csv file should be saved as one output.csv file. This allows fast batch processing later [0.25 pt]

Text

Description automatically generated

Table

Description automatically generated