UNIVERSITY OF ARKANSAS

CSCE 46103/56103: Introduction to Artificial Intelligence

Homework #1

Submission Deadline: 11:59 PM, September 09th, 2024

Student ID: 010974718

Student Name: Blake Williams

Submission Instruction

Read all the instructions below carefully before you start working on the homework, and before submission.

- Your submission contains 2 parts: PDF file and source code folder. All is zipped in a single file. The zip file should be named with your student ID.
- Your submission will be as follow:

```
0123456.zip
          0123456.pdf
          Source_code
                    Question_2
                    Question_3
                    Question_4
```

- If you have questions, please contact Gladys Gawugah ggawugah@uark.edu; gladysgawugah@gmail.com

I. Question 1: Linear Algebra [20 pts]

Problem: You are given a matrix $\mathbf{A} \in R^{3\times 3}$ and matrix $\mathbf{B} \in R^{3\times 3}$.

$$\mathbf{A} = \begin{bmatrix} 1 & 3 & 4 \\ 1 & 3 & 5 \\ 1 & 3 & 6 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 7 & 1 & 9 \\ 2 & 2 & 5 \\ 1 & 3 & 4 \end{bmatrix} \tag{1}$$

(a) **[2 pts]** What is the matrix multiplication result of **AB**?

17	19	40
18	22	44
19	25	48

(b) **[3 pts]** How many column vectors in **A** are independent?

2

(c) **[3 pts]** Column space of **A** spans which one: a line, a plane or R^3?

plane

(d) **[3 pts]** How many column vectors in **B** are independent?

3

(e) **[3 pts]** Column space of **B** spans which one: a line, a plane or R^3?

R^3

(f) **[3 pts]** How many column vectors in **AB** are independent?

2

(g) **[3 pts]** Column space of **AB** spans which one: a line, a plane or R^3?

plane

Question 2: BFS, DFS[20 pts]

You are given an m x n matrix (m,n are the number of rows and columns, respectively), and the matrix is a zero-based index (the index of row and column start from 0). Each cell of the matrix will have a number. It will be either 0 or 1, 0 means you cannot access this cell, and 1 means you can access this cell. You are in the cell (0,0), your mission is to find the path that reaches the cell (m,n). It should be noted that at each step, you are just allowed to move to adjacent cells. Particularly, if you are in the cell (x,y), you can move to either cell (x-1,y) or (x,y-1) or (x+1,y) or (x,y+1). Also, you have to make sure that you are not going out of the matrix and you cannot access any cells that contain 0. In addition, you are just allowed to go to each cell at most one time.

Example:

1	1	1	0	1
0	0	1	1	0
0	1	1	1	1
1	1	0	1	0
0	1	0	1	1

Output: (0,0)-→(0,1)-→(0,2)-→(1,2)-→(2,2)-→(2,3)-→(3,3)-→(4,3)-→(4,4) Notes: It may have more than 1 output; in this case, you can print one of either of them. If you cannot find any path, please print -1.

(a) [10 pts] You are required to implement DFS to solve the problem above.

```
def DFS(m,n,matrix):

        visited = [[False for _ in range(n)] for _ in range(m)]
        stack = deque()

        stack.append((0, 0, [(0, 0)]))  # (x, y, path)
        visited[0][0] = True

        while stack:
            x, y, path = stack.pop()

            if x == m - 1 and y == n - 1:
                print_path(path)
                return

            for i in range(4):
                new_x, new_y = x + dx[i], y + dy[i]

                if is_valid(new_x, new_y, m, n, matrix, visited):
                    visited[new_x][new_y] = True
                    stack.append((new_x, new_y, path + [(new_x, new_y)]))

        print("-1")
```

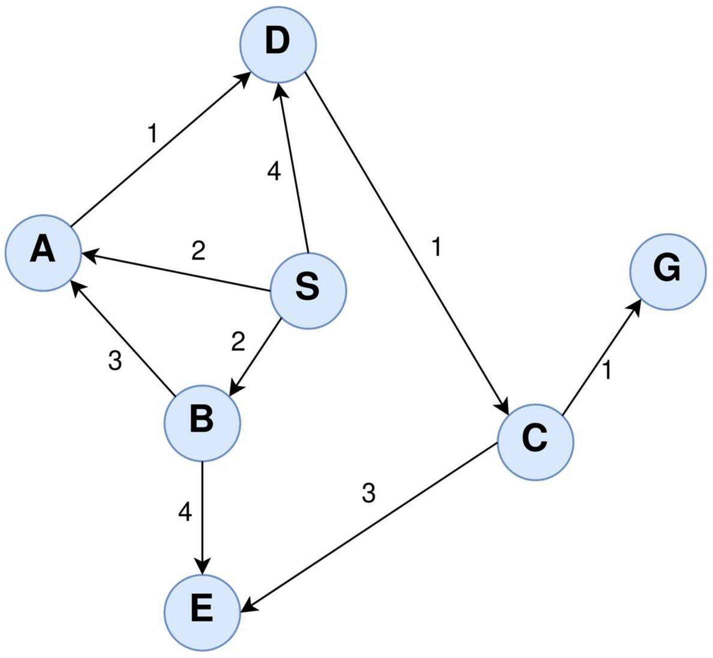
(b) [10 pts] You are required to implement BFS to solve the above problem.

```
def BFS(m, n, matrix):

        visited = [[False for _ in range(n)] for _ in range(m)]
        stack = deque()

        stack.append((0, 0, [(0, 0)]))  # (x, y, path)
        visited[0][0] = True

        while stack:
            x, y, path = stack.pop()

            if x == m - 1 and y == n - 1:
                print_path(path)
                return

            for i in range(4):
                new_x, new_y = x + dx[i], y + dy[i]

                if is_valid(new_x, new_y, m, n, matrix, visited):
                    visited[new_x][new_y] = True
                    stack.append((new_x, new_y, path + [(new_x, new_y)]))

        print("-1")
```

NOTE: SEE SOURCE CODE FOR FULL SOLUTION (q2.py)

Question 3: DFS, BFS, UCS [30 pts]

You are given a graph consisting of **seven** states and the costs of connection between them for each state. Please refer to the lecture for example answers. You are asked to take the cycle/looping matter into account.


$$\begin{tikzcd}
 & D \arrow[drr, "1"] & & & \\
A \arrow[ur, "1"] & & S \arrow[ll, "2"'] \arrow[ul, "4"'] \arrow[dl, "2"] & & G \\
 & B \arrow[ul, "3"] \arrow[d, "4"] & & C \arrow[ur, "1"'] \arrow[dll, "3"] & \\
 & E & & &
\end{tikzcd}$$

Run q3-4.py, Select Graph 2, and input S then G to get output for DFS, BFS, and UCS

(a) **[10 pts]** Perform Depth-first Search (DFS) from S to G.

(a.1) **[2.5 pts]** Return search table of DFS.

'Node': 'S', 'Parent': None, 'Path': ['S']

'Node': 'D', 'Parent': 'S', 'Path': ['S', 'D']

'Node': 'C', 'Parent': 'D', 'Path': ['S', 'D', 'C']

'Node': 'G', 'Parent': 'C', 'Path': ['S', 'D', 'C', 'G']

(a.2) **[2.5 pts]** What is the path returned?

['S', 'D', 'C', 'G']

(a.3) **[2.5 pts]** What is the node expansion order?

['S', 'D', 'C', 'G']

(a.4) **[2.5 pts]** What is the path cost of the returned path?

6

(b) **[10 pts]** Perform Breadth-first Search (BFS) from S to G:

(b.1) **[2.5 pts]** Return search table of DFS.

'Node': 'S', 'Parent': None, 'Path': ['S']

'Node': 'D', 'Parent': 'S', 'Path': ['S', 'D']

'Node': 'A', 'Parent': 'S', 'Path': ['S', 'A']

'Node': 'B', 'Parent': 'S', 'Path': ['S', 'B']

'Node': 'C', 'Parent': 'D', 'Path': ['S', 'D', 'C']

'Node': 'D', 'Parent': 'A', 'Path': ['S', 'A', 'D']

'Node': 'A', 'Parent': 'B', 'Path': ['S', 'B', 'A']

'Node': 'E', 'Parent': 'B', 'Path': ['S', 'B', 'E']

'Node': 'G', 'Parent': 'C', 'Path': ['S', 'D', 'C', 'G']

(b.2) **[2.5 pts]** What is the path returned?

['S', 'D', 'C', 'G']

(b.3) **[2.5 pts]** What is the node expansion order?

['S', 'D', 'A', 'B', 'C', 'E', 'G']

(b.4) **[2.5 pts]** What is the path cost of the returned path?

6

(c) **[10 pts]** Perform Uniform-cost Search (UCS) from S to G:

(a.1) **[2.5 pts]** Return search table of UCS.

'Node': 'S', 'Parent': None, 'Cost': 0, 'Path': ['S']

'Node': 'A', 'Parent': 'S', 'Cost': 2, 'Path': ['S', 'A']

'Node': 'B', 'Parent': 'S', 'Cost': 2, 'Path': ['S', 'B']

'Node': 'D', 'Parent': 'A', 'Cost': 3, 'Path': ['S', 'A', 'D']

'Node': 'C', 'Parent': 'D', 'Cost': 4, 'Path': ['S', 'A', 'D', 'C']

'Node': 'D', 'Parent': 'S', 'Cost': 4, 'Path': ['S', 'D']

'Node': 'G', 'Parent': 'C', 'Cost': 5, 'Path': ['S', 'A', 'D', 'C', 'G']

(a.2) **[2.5 pts]** What is the path returned?

['S', 'A', 'D', 'C', 'G']

(a.3) **[2.5 pts]** What is the node expansion order?

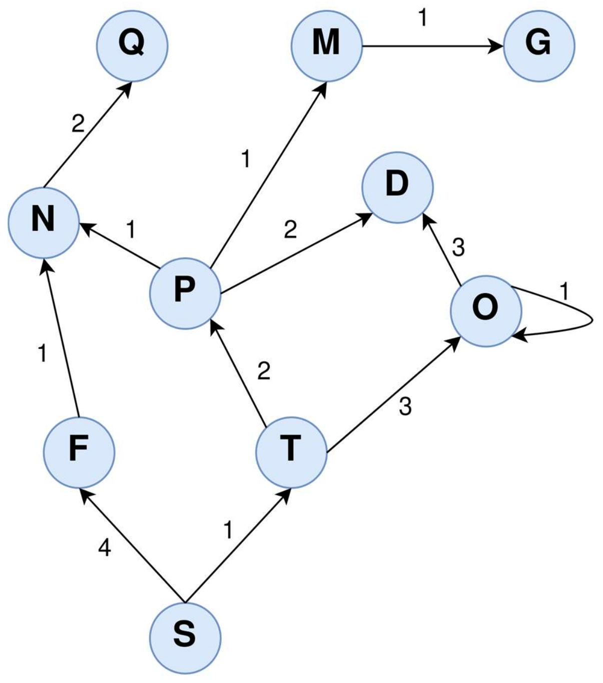
['S', 'A', 'B', 'D', 'C', 'G']

(a.4) **[2.5 pts]** What is the path cost of the returned path?

5

Question 4: Uninformed Search [30 pts]

You are given a graph consisting of **ten** states and the costs of connection between them for each state. You are asked to take the cycle/looping matter into account.



Run q3-4.py, Select Graph 1, and input S then G to get output for DFS, BFS, and UCS

(a) Perform Depth-first Search (DFS) from *S* to *G*:

(a.1) **[2.5 pts]** Return search table of DFS .

'Node': 'S', 'Parent': None, 'Path': ['S']

'Node': 'F', 'Parent': 'S', 'Path': ['S', 'F']

'Node': 'N', 'Parent': 'F', 'Path': ['S', 'F', 'N']

'Node': 'Q', 'Parent': 'N', 'Path': ['S', 'F', 'N', 'Q']

'Node': 'T', 'Parent': 'S', 'Path': ['S', 'T']

'Node': 'P', 'Parent': 'T', 'Path': ['S', 'T', 'P']

'Node': 'N', 'Parent': 'P', 'Path': ['S', 'T', 'P', 'N']

'Node': 'Q', 'Parent': 'N', 'Path': ['S', 'T', 'P', 'N', 'Q']

'Node': 'M', 'Parent': 'P', 'Path': ['S', 'T', 'P', 'M']

'Node': 'G', 'Parent': 'M', 'Path': ['S', 'T', 'P', 'M', 'G']

(a.1) **[2.5 pts]** What is the path returned?

['S', 'T', 'P', 'M', 'G']

(a.2) **[2.5 pts]** What is the node expansion order?

['S', 'F', 'N', 'Q', 'T', 'P', 'M', 'G']

(a.3) **[2.5 pts]** What is the path cost of the returned path?

5

(b) Perform Breadth-first Search (BFS) from S to G:

(a.1) [2.5 pts] Return search table of BFS

'Node': 'S', 'Parent': None, 'Path': ['S']

'Node': 'F', 'Parent': 'S', 'Path': ['S', 'F']

'Node': 'T', 'Parent': 'S', 'Path': ['S', 'T']

'Node': 'N', 'Parent': 'F', 'Path': ['S', 'F', 'N']

'Node': 'P', 'Parent': 'T', 'Path': ['S', 'T', 'P']

'Node': 'O', 'Parent': 'T', 'Path': ['S', 'T', 'O']

'Node': 'Q', 'Parent': 'N', 'Path': ['S', 'F', 'N', 'Q']

'Node': 'N', 'Parent': 'P', 'Path': ['S', 'T', 'P', 'N']

'Node': 'M', 'Parent': 'P', 'Path': ['S', 'T', 'P', 'M']

'Node': 'D', 'Parent': 'P', 'Path': ['S', 'T', 'P', 'D']

'Node': 'O', 'Parent': 'O', 'Path': ['S', 'T', 'O', 'O']

'Node': 'D', 'Parent': 'O', 'Path': ['S', 'T', 'O', 'D']

'Node': 'G', 'Parent': 'M', 'Path': ['S', 'T', 'P', 'M', 'G']

(b.1) **[2.5 pts]** What is the path returned?

['S', 'T', 'P', 'M', 'G']

(b.2) **[2.5 pts]** What is the node expansion order?

['S', 'F', 'T', 'N', 'P', 'O', 'Q', 'M', 'D', 'G']

(b.3) **[2.5 pts]** What is the path cost of the returned path

5

(c) Perform Uniform-cost Search (UCS) from S to G:

(a.1) [2.5 pts] Return search table of UCS

'Node': 'S', 'Parent': None, 'Cost': 0, 'Path': ['S']

‘Node': 'T', 'Parent': 'S', 'Cost': 1, 'Path': ['S', 'T']

'Node': 'P', 'Parent': 'T', 'Cost': 3, 'Path': ['S', 'T', 'P']

'Node': 'F', 'Parent': 'S', 'Cost': 4, 'Path': ['S', 'F']

'Node': 'O', 'Parent': 'T', 'Cost': 4, 'Path': ['S', 'T', 'O']

'Node': 'M', 'Parent': 'P', 'Cost': 4, 'Path': ['S', 'T', 'P', 'M']

'Node': 'N', 'Parent': 'P', 'Cost': 4, 'Path': ['S', 'T', 'P', 'N']

'Node': 'N', 'Parent': 'F', 'Cost': 5, 'Path': ['S', 'F', 'N']

'Node': 'O', 'Parent': 'O', 'Cost': 5, 'Path': ['S', 'T', 'O', 'O']

'Node': 'D', 'Parent': 'P', 'Cost': 5, 'Path': ['S', 'T', 'P', 'D']

'Node': 'G', 'Parent': 'M', 'Cost': 5, 'Path': ['S', 'T', 'P', 'M', 'G']

(c.1) [2.5 pts] What is the path returned?

['S', 'T', 'P', 'M', 'G']

(c.2) **[2.5 pts]** What is the node expansion order?

['S', 'T', 'P', 'F', 'O', 'M', 'N', 'D', 'G']

(c.3) **[2.5 pts]** What is the path cost of the returned path?

5