Assignment 2
CSC 520 Artificial Intelligence

Due February 14th, 2022

This assignment consists of two questions which involve written answers and code. In order to complete the assignment you must submit a written report in pdf form detailing your answers to the questions as well as your code. As discussed in class all work must be your own. You may not use third party libraries or example code to complete the assignment. All reports must be clear and well written. All code must be clear, readable, and well-commented. Please include a README with instructions on how to run the code.

Question 1

You're in a remote town [S] with your car and are low on gas. Gas stations are present in the nearby towns [G1 and G2] and reaching the nearest one is your Goal. You have access to a map that include roads between cities and the distances between each pair of cities with a road between them (available in the . You have to plan to solve this issue using A* heuristic search. One of your friends, who has a good knowledge of roads and has a fair idea of the terrain, can guess the approximate distance of the Goals (nearest petrol pumps) from any town. The distances calculated by your friend to the goal state are the 'heuristic distances'. Distances of paths to the immediately further towns are the 'actual distances. Write a program to arrive at the final path that the algorithm would take using the A* algorithm. Your code should output the fina; path and the total cost of getting from the current city to your goal destination (total distance).

The distances calculated by your friend to the goal state are the 'heuristic distances'. Distances of paths to the immediately further towns are the 'actual distances'.

The test case files are present in the folder **Q1**.

Write a program to arrive at the final path that the algorithm would take using the A* algorithm. Your code should output the path and the distance of the path till that step.

Input: Refer file - 'case1.txt'

In the first table you are given the list of towns and the corresponding Heuristic values.
Second table comprises the actual path distances from one town to another.

Example:
Initial State: S
3 paths are possible : SA, SB, SC

A* Heuristic values:
SA - 15 + 30 = 45
SB - 2 + 40 = 42
SC - 15 + 10 = 25

Queue = [[SA, 45], [SB, 42], [SC, 25]]
Choose values with the lowest A* Heuristic value: [SC, 25].

Continue the same steps till a goal state is reached.

Expected Output:
['SBDAEG1', 45]

Execution Instructions :

1. Please make sure that your code is executable on running the command :

For Python:
python3 q1.py <filename>
Example command : *python3 q1.py case1.txt*

For Java:
javac q1.java
java q1 <filename>

2. Write the output of the program to a text file in the **Q1/Solutions** folder with the naming convention - *<filename>_solution.txt*.

Question 2

The following questions have been designed so that you can apply the search algorithms you learnt in class to help Harry escape two mazes. In both the scenarios detailed below, Harry (represented by H) is trapped in a maze with empty spaces (represented by 0) and walls (represented as #).

0.1 Scenario 1: (Use DFS and BFS)

The only way Harry can get out of this maze is if he can reach the teleporting cell (represented as T). Harry can traverse through the maze on his magic broom but it is malfunctioning. The broom can go through the empty spaces by flying straight, back, left or right, but it won't stop flying until hitting a wall. When it hits a wall, it lets Harry choose the next direction.

Harry has a navigator to help him plan out his moves ("after all programming is the closest thing we have to magic"). Can you write two separate programs (implementing the DFS and BFS algorithm) for the navigator so that Harry can find his way to the teleporting cell?

The testcase files are present in the folder **Q2/Mazes/DFS**. Given the m x n maze, return the final path Harry should follow to escape from the maze if he can, otherwise return Not possible. You are required to parse the given maze file to identify the starting position of Harry.

Consider the following as the preference of direction when traversing the maze: ***left, right, up, down***.

Following are 2 sample mazes:

#	#	#	#	#	#	#
#	0	0	#	0	H	#
#	0	0	T	0	0	#
#	0	0	0	#	0	#
#	#	#	0	#	#	#
#	0	0	0	0	0	#
#	#	#	#	#	#	#

Table 1: Maze 1

You are expected to return a comma separated tuple of the row index and column index of the nodes in the order they were visited.

Expected Output : (1, 5), (1, 4), (2, 4), (2, 1), (1, 1), (1, 2), (3, 2), (3, 1), (3, 3), (2, 3)

#	#	#	#	#	#	#
#	0	0	#	0	H	#
#	0	0	0	0	0	#
#	0	0	0	#	0	#
#	#	#	T	#	#	#
#	0	0	0	0	0	#
#	#	#	#	#	#	#

Table 2: Maze 2

Expected Output : Not possible

0.2 Scenario 2: (Use A*)

The only way Harry can get out of this maze is if he can collect all the keys (represented as K) and reach the door (represented by D). The location of the keys is known to Harry. He can traverse through the maze by walking left, right, straight or back.

Harry has a navigator to help him plan out his moves. Assume that the energy Harry has to spend to move from one cell to another is 1 (since he doesn't have his magic broom now). Can you write a program for the navigator so that Harry can collect all the keys and reach the door by spending the least amount of energy?

The testcase files are present in the folder **Q2/Mazes/ASTAR**. Given the m x n maze, if it is possible for Harry to collect all the keys and reach the door, return the minimum number of steps taken, the order in which the keys were collected (return the position of the cell in which the keys are present) and the path taken, else return -1.

You are required to parse the given maze file to identify the location of all the keys to devise your plan. The starting position of Harry is always (0,0) and the door is always at the rightmost column in the last row of the maze.

Define a heuristic function that you can use for this use case. Is your heuristic consistent and admissible? Explain (you don't have to prove either admissibility or consistency, an intuitive explanation suffices). Please make sure to mention the heuristic function used in the PDF that would be submitted and also in the README for the code.

Consider the following as the preference of direction when traversing the maze: ***right, down, left and up****.*

Following are 2 sample mazes:

H	0	0	0	0	0
0	#	K	0	0	0

0	#	0	0	#	K
0	#	0	K	#	D

Table 3: Maze 1

The order in which the keys are found for this maze is : (1, 2), (3, 3), (2, 5). Harry should first collect the key at position (1, 2), then at (3,3) and then (2,5).

Number of moves taken to collect all the keys and reach the door : 12

The path taken to collect all the keys and reach the door is : (0,0) → (0,1) → (0.2) → (1,2) → (2,2) → (3,2) → (3,3) → (2,3) → (1,3) → (1,4) → (1,5) → (2,5) → (2,6)

Expected Output:

1. Order : (1, 2), (3, 3), (2, 5)
2. Number of moves : 12
3. Path : (0,0),(0,1),(0.2),(1,2),(2,2),(3,2),(3,3),(2,3),(1,3),(1,4),(1,5),(2,5),(2,6)

H	0	0	0	0	0
0	#	K	0	0	0
0	#	#	#	#	K
0	#	0	K	#	D

Table 4: Maze 2

The key at position (3,3) is not reachable as it is surrounded by obstacles in all directions. In this case, Harry cannot unlock the door even if he reaches it as he has not collected one key. In such scenarios, return the list of keys you were able to collect - (1, 2),(2, 5) and -1 for number of moves and “Not possible” as path.

Expected Output:

1. Order : (1, 2),(2, 5)
2. Moves : -1
3. Path : Not possible

Execution Instructions :

1. Please make sure that your code is executable on running the command :
   For Python:
   python3 q2.py <maze filename> <dfs / astar>
   Example command : *python3 q2.py Maze1.txt astar*

   For Java:
   javac q2.java
   java q2 <maze filename> <dfs / astar>

2. Write the output of the program to a text file in the **Q2/Solutions** folder with the naming convention - <*Maze filename*>*_solution.txt*. Answers to Scenario 1, Scenario 2 must be in the **Q2/Solutions/DFS, Q2/Solutions/ASTAR** folders respectively.

3. Sample solution files for the mazes above have been added to the **Q2/Solutions** folder.

For both, Scenario 1 and Scenario 2, obtain the number of paths expanded for each test case. Store them in a table and compare the results. What can you infer from it?