COMP 237 - Online lab assignment "Search"

Due date: End of week #5

Purpose:
The purpose of this Lab assignment is to:

1. To get hands-on experience of applying uninformed search algorithms to solve a business problem.
2. To examine and recommend best heuristics to solve a gamming problem using informed search algorithms.

Pre-requisite to carrying out the assignment:

1. download from the course shell the following Python scripts, examine and test:
    a. Depth first Search (zipped folder)
    b. Breadth first Search (zipped folder)
    c. Greedy best first search
    d. A* star search
2. Go through and watch all "Uninformed & Informed" lecture and lab tutorials related to modules #3 & 4 to understand the concepts and how the code works.

General Instructions:
Be sure to read the following general instructions carefully:

1. This assignment must be completed individually by all the students.
2. Only provide the requested screenshots and make sure to have a complete screenshot, partial screenshots will not earn any marks.
3. You will have to provide a **demonstration video for your solution** and upload the video together with the solution on **eCentennial** through the assignment link. See the **video recording instructions** at the end of this document.
4. In your 5-minute demonstration video you should explain your solution clearly, going over the main code blocks and the purpose of each module/class/method also demoing the execution of exercises #1 & 2. Any submission without an accompanying video will lose 70% of the grade.
   Submission:

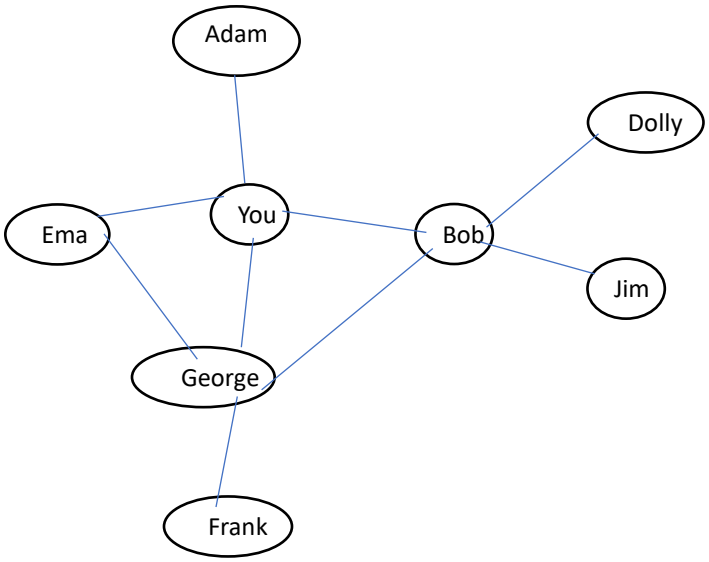
1. for each exercise that requires code, please create a zipped folder project and include all project python scripts and screenshot of output. Name the folder "Exercise#X_firstname", where X is the exercise number and firstname is your first name. (In total 3 zipped packages)
2. For all questions that require written or graphic response create one-word document and indicate the exercise number and then state your response. Name the document "Written_responses_firstname", where firstname is your firstname. (In total one word or pdf document).
3. All submissions need to be accompanied with a recorded demonstration video not to exceed 7 minutes in length.

Note: In all coding amendments, make sure to comment your code with explanations.

Assignment - exercises:

1. Exercise # 1 (Classmate search) (35 marks)

   You have the following undirected graph that illustrates relationships amongst students in your class, replace “You” by your first name:



Write a script that would use Breadth first search (BFS) to allow for the students’ introductions to each other with the least number of intermediate students for the introduction, i.e. find the shortest path.

In developing the script please consider the following:

1) Store the relationships in a dictionary structure, save the structure as a separate module and import that structure into your script. For example: graph = {'Adam': ['You'],

'Bob': [You','Dolly', ‘Jim’, ’George’],

……..,

……}

2) You may use any of the python data structures queues, stacks, priority queues, or create your own using python native data structures such as lists, dictionaries….etc.

3) You will have to keep track of visited nodes, the frontier, the current node and the final path. You can reuse any of lab scripts or completely build your own.

4) You will have to create a function named BFS_firstname where firstname is your firstname that will contain all the necessary logic to build the algorithm.

5) The function in point four above should accept i.e. have three arguments that can be changed, as follows:

1. graph_name.

2. (Start/initial) student name who wishes to connect.
3. (End/goal) student name of target person.

For example, if you store the relationships under a structure named “GraphX” and “Ema” wishes to get introduced to “Bob” then you would call the function from the main as follows:

BFS_firstname (“GraphX”, “Ema”, “Bob”)

6) If a relationship cannot be established then your code should give an appropriate apology message.

7) If any of the passed arguments do not exist then an appropriate message should be returned.

8) If any names passed as arguments do not exist on the graph then you should display an appropriate message.

9) As output print the final path and the tree traversed, you can use Graphviz to illustrate your results, or print out.

Run the script on the basis that Dolly needs to get introduced to you (replace you by your first name).

Then rerun on the basis that Frank needs to get introduced to Dolly.

2. Exercise # 2: Map search (40 marks)

For the location map search example explained in the Greedy best first search and A* algorithm, change the search algorithm to follow the uniform cost algorithm (UCS). Run the new code and compare the results of UCS versus Greedy & A*, in terms of the levels required and the number of nodes expanded.

Write in your own words a short analysis of the results you obtained in the “Written_responses_firstname” document.

3. Exercise # 3: A* (25 marks) does not require code

Given an initial state of an 8-puzzle problem and final state to be reached as follows:

2	6	
4	5	1
7	3	8

Initial state

2	6	1
	4	5
7	3	8

Final state

Assuming our agent needs to figure out the shortest path from the initial state to the final state using the A* algorithm and the heuristic function **h(n)** used is: **“The number of misplaced tiles”.** The cost function **g(n)** is: **“the depth”** as we expand the tree towards a solution. Also the agent can move one step at a time and either horizontally or vertically i.e. no diagonal movements.

Answer to the following questions:

a) Complete the tree search showing all possible states that the agent will investigate to reach to the final state, indicating the values of h(n), g(n) and f(n).

b) Is this a good heuristic for the problem, explain why.

c) Suggest another heuristic, explain why you think it would be a good heuristic.

------ End of Exercises ------

Demonstration Video Recording

Please record a short video (max 5-7 minutes) to explain/demonstrate your assignment solution. You may **use the Windows 10 Game bar** to do the recording:

1. Press the Windows key + G at the same time to open the Game Bar dialog.
2. Check the "Yes, this is a game" checkbox to load the Game Bar.
3. Click on the Start Recording button (or Win + Alt + R) to begin capturing the video.
4. Stop the recording by clicking on the red recording bar that will be on the top right of the program window.

(If it disappears on you, press Win + G again to bring the Game Bar back.)

You'll find your recorded video (MP4 file), under the Videos folder in a subfolder called Captures.

Submit the video together with your solution.

Alternatively, you can use any other freely available software. One option is screencast-O-matic free version at:
https://screencast-o-matic.com/