CS 156 – Introduction to AI
Assignment 2

Instructions:

- Submission type: 3 files; a PDF file and two python scripts named “A2a_ToDo.py” and “A2b_ToDo.py”
- Maximum points: 100
- Due: Jun 18th, 11:59 PM
- Do not use AI tools to generate content for this assignment. You can use it for research, but all material submitted as answer should be your own. You should be ready to explain the logic and reasoning behind your answers.

Section 1 – 40 Points

1. Problem Formulation
   Give a complete problem formulation for each of the following problems. Choose a formulation that is precise enough to be implemented. Problem formulation must consist of; **Initial State, Goal test, Successor/Action function, Cost function.**

   a) (5 points) There are six glass boxes in a row, each with a lock. Each of the first five boxes holds a key unlocking the next box in line; the last box holds a fruit. You have the key to the first box, and you want the fruit.
   b) (5 points) There is an n × n grid of squares, each square initially being unpainted floor. You start standing on an unpainted floor square and can either paint the square under you or move onto an adjacent unpainted floor square. You want the whole floor painted.
   c) (5 points) A container ship is in port, loaded high with containers. There are 13 rows of containers, each 13 containers wide and 5 containers tall. You control a crane that can move to any location above the ship, pick up the container under it, and move it onto the dock. You want the ship unloaded.

2. Your goal is to navigate a robot out of a maze. The robot starts in the center of the maze facing north. You can turn the robot to face north, east, south, or west. You can direct the robot to move forward a certain distance, although it will stop before hitting a wall. For sub-questions a, b and c, you have to answer with problem formulation consisting of; **Initial State, Goal test, Successor/Action function, Cost function** which is updated to suit each sub-question.

   a) (4 points) Formulate this problem. How large is the state space?
   b) (4 points) In navigating a maze, the only place we need to turn is at the intersection of two or more corridors. Reformulate this problem using this observation. How large is the state space now?

c) (4 points) From each point in the maze, we can move in any of the four directions until we reach a turning point, and this is the only action we need to do. Reformulate the problem using these actions. Do we need to keep track of the robot's orientation now?

d) (3 points) In our initial description of this problem, we already abstracted from the real world, restricting actions and removing details. List three such simplifications we made.

3. Search Strategies (10 points)
   Briefly explain, compare and contrast the Breadth First Search, Depth First Search, Iterative Deeping Search and the AStar Search in the context of search trees.

Section 2 – 60 points

Part A Instructions:

a) Use the files in Assignment2.zip.
b) A2a_Base.py file contains the class definition for Graph, GraphProblem and a hardcoded state transition dictionary for the graph example we discussed in class. DO NOT EDIT THIS FILE.
c) A2a_ToDo.py consists of the skeleton code, which should be completed. Your implementation goes in this file.
d) A2a_Tests.py consists of test cases for the assignment. DO NOT EDIT THIS FILE. To run this file in PyCharm, open the file, select "Run" from the menu bar, and select "run pytest in A2a_Tests.py".
e) You only need to submit the A2a_ToDo.py

1. (15 points) Implement the breadth first search by completing the function

   ```
   def breadth_first_graph_search(problem):
   ```

   each test case must be passed in A2a_Tests.py in test_breadth_first_graph_search()

2. (15 points) Implement the depth first search by completing the function

   ```
   def depth_first_graph_search(problem):
   ```

   each test case must be passed in A2a_Tests.py in test_depth_first_graph_search()

Part B Instructions:

a) A2b_Base.py file contains the class definition for Graph, GraphProblem and a hardcoded state transition dictionary for the graph example we discussed in class. DO NOT EDIT THIS FILE.

b) A2b_ToDo.py consists of the skeleton code, which should be completed. Your implementation goes in this file.
c) A2b_Tests.py consists of test cases for the assignment. DO NOT EDIT THIS FILE. To run this file in PyCharm, open the file, select “Run” from the menu bar, and select “run pytest in A2b_Tests.py”.
d) You only need to submit the A2b_ToDo.py

3. (15 points) Implement the AStar search by completing the function

   ```
   def astar_search(problem, h=None):
   ```

   NOTE: Your code must print out f(n) = h(n) + path_cost values for nodes as they are explored/expanded.

4. (15 points) Implement the AStarGraph sub-class which inherits the GraphProblem class. Add a heuristic function h which computes the straight line distance between cities using Romania_map.locations.
   NOTE : Your implementation must print out the h values as and when they are computed.

   each test case must be passed in A2b_Tests.py