CS 534 Homework Assignment 2 (100 Points)
Due: 11:59 p.m. on 06/12/2022

Homework Objective: The goal of this assignment is to help you master the materials of Lecture 1, 2, and 3.

Homework Deliverables: Submit your answers and solutions in a ***zip*** file that includes all the write-ups in a ***pdf*** file and your ***python codes*** to Canvas.

Project Questions:

Question 1.

Part a: Use the **agents4e.py** and its **ReflexVacuumAgent()** function provided to develop and implement our vacuum cleaner discussed in Week 1. Run the environment with this agent for all possible initial dirt configurations and agent locations. When we run your code, your program will generate the correct action with the corresponding input percept sequence.

Part a: Use the **agents4e.py** and its **SimpleReflexAgentProgram()** function provided to develop and implement our vacuum cleaner discussed in Week 1. Run the environment with this agent for all possible initial dirt configurations and agent locations. When we run your code, your program will generate the correct action with the corresponding input percept sequence.

<u>Notes</u>:
1. The implementation should be in Python.
2. Output the sequence of actions for the environment and initial locations described in Figure 2.2, 2.3, 2.8, 2.9, and 2.10 of your AI textbook, the 4th edition.

Question 2. Suppose two friends live in different cities on a map, such as the Romania map shown in Figure 3.1. On every turn, we can simultaneously move each friend to a neighboring city on the map. The amount of time needed to move from city i to neighbor j is equal to the road distance *D(i, j)* between the cities, but on each turn the friend that arrives first must wait until the other one arrives (and calls the first on his/her cell phone) before the next turn can begin. We want the two friends to meet as quickly as possible.

1. Write a detailed formulation for this search problem. (You will find it helpful to define some formal notation here.)
2. Let *D(i, j)* be the straight-line distance between cities *i* and *j*. Which of the following heuristic functions are admissible? (i) *D(i, j)*; (ii) *2 · D(i, j)*; (iii) *D(i, j)/2*.
3. Are there completely connected maps for which no solution exists?
4. Are there maps in which all solutions require one friend to visit the same city twice?

<u>Note</u>: Each answer needs to be explained.

Question 3. In this question, we explore the use of local search methods to solve Traveling Salesperson Problems (TSPs) that are described on the Page 70 of your textbook. You can also refer to this website, https://www.tutorialspoint.com/design_and_analysis_of_algorithms/design_and_analysis_of_algorithms_travelling_salesman_problem.htm, for a reference as a starting point.

a. Implement and test a hill-climbing algorithm to solve TSPs.
b. Repeat part (a) using a genetic algorithm instead of hill climbing.
c. Compare the results with the above two solutions.

When we run your code using either hill-climbing or genetic algorithm, each program algorithm generates its own shortest possible route, respectively, that he visits each city exactly once and returns to the origin city.

Notes: Both algorithms can be found in **search.py**. The implementation should be in Python.

Question 4. Describe how the minimax and alpha–beta algorithms change for two-player, non-zero-sum games in which each player has a distinct utility function and both utility functions are known to both players. If there are no constraints on the two terminal utilities:

a. Is it possible for any node to be pruned by alpha–beta?
b. What if the player's utility functions on any state differ by at most a constant *k*, making the game almost cooperative?
c. What if the player's utility functions on any state sum to a number between constants $-k$ and k, making the game almost zero-sum?

Note: Each answer needs to be explained.

Question 5. Consider the problem of placing k knights on an $n \times n$ chessboard such that no two knights are attacking each other, where k is given and $k \leq n^2$.

a. Choose a CSP formulation. In your formulation, what are the variables?
b. What are the possible values of each variable?
c. What sets of variables are constrained, and how?
d. Now consider the problem of putting *as many knights as possible* on the board without any attacks. Explain how to solve this with local search by defining appropriate ACTIONS and RESULT functions and a sensible objective function.

Notes: Each answer needs to be explained.

Grading Criteria: Your answers must be complete and clear.