CSE 471 HW 1: Intro to AI and Uninformed Search

**Due on Monday January 27th at midnight** *(submit using Canvas, show your work to get full credit)*

Name: Addison Nou

ASU ID: 1211275907

Email: arnou@asu.edu

**2.3:**

a. False; a rational agent is fully capable of acting rationally even if it can only receive partial information. An example of this is a roomba – it does not need to know the entire layout of the room to make a rational decision.

b. True; anything that requires memory would make a reflex agent insufficient in making rational decisions.

c. True; environments in which there is only one action to take.

d. False; the input to an agent program is the percept history, while the input to the agent function is just the current percept.

e. False; if an agent function is infinitely large, any machine will run out of memory before completing the function

f. True; again, take for example the environment in which there is only a single action to take.

g. True; if the environments are similar enough, agent function may still be rational.

h. False; whether or not it is observable does not affect the actions that the agent takes.

i. False; a game of poker also relies on luck, which a rational agent has no control over.

**2.4:**

a. Playing Soccer:

- Performance Measure = Win/Lose

- Environment = Soccer field

- Actuators = Feet, legs, upper body, head  
 - Sensors = Eyes, ears

b. Exploring the subsurface oceans of Titan

- Performance Measure = Amount of area explored

- Environment = Subsurface oceans of Titan

- Actuators = Ship accelerator, decelerator

- Sensors = Camera, probe, sonar

c. Shopping for used AI books on the internet

- Performance Measurement = Number of books found

- Environment = Internet

- Actuators = Keyboard, mouse

- Sensors = computer screen, browser interfaces

d. Playing tennis against a wall

- Performance Measurement = Number of tennis ball bounces in a row

- Environment = Next to a wall

- Actuators = Tennis racket, arms, legs

- Sensors = Eyes, ears

e. Performing a high jump

- Performance Measurement = Jump height

- Environment = High jump setup

- Actuators = Legs, body

- Sensors = Eyes

f. Knitting a sweater

- Performance Measurement = How well the sweater was knitted

- Environment = A chair/worktable

- Actuators = Sewing machine, hands, needle

- Sensors = Eyes

g. Bidding on an item at an auction

- Performance Measurement = Acquiring the item

- Envrionment = Auction website/house

- Actuators = Mouse, hands, bidding sign

- Sensors = Eyes

**3.2:**

a. Total number of states is 4x, where x is the number of ‘blocks’ in the maze’

b. Total number of states is 4I + 2C, where I is the number of intersections and C is the number of corridors

c. The total number of states is I, where I is the number of turning points. This is because the AI only cares about if we need to turn or not, and that turn is the only action that it makes. Thus, we do not care about orientation.

d.

- We can only move in 4 directions instead of diagonally.

- It does not have any case for reaching dead ends

- There are no cases in which the robot improperly senses walls or gets stuck

**3.3:**

a.

States: <i, j> where <i, j> i and j are all possible cities, and the pair is the location of each friend

Initial state: Any pair <i, j>

Transition model: Distance between moving from one city to another

Goal: <i, j> where i and j are equal to each other

Path cost: Maximum distance between <i, y> and <x, j>, where x and y are adjacent cities. In other words, cost = max(d(i, y), d(x, j)).

b. D(i, j)/2, so the distance each friend travels is the same.

c. If placed in a map with only two cities, the friends will continually swap places, and will never meet up.

d. Yes, there are maps that such a solution is necessary. For example, if there is a set of 4 nodes (A, B, C, D) where A is connected to B is connected to D, and C is connected to B and D, and the friends start at A and B, one will be required to double back to meet up with their friend.

**3.5:** Assuming the first queen can take ‘n’ spaces, a lowerbound case would be that the second queen can take n-3 spaces in the second column, n-6 in the third column, and so on and so forth. The mathematical expression for this would be (n \* n-3 \* n-6) = 3√n!

**3.6:**

a.

- Initial state: Colorless region

- Successor: Color a region a different color than the region adjacent

- Path cost: The time to color all sections of the map

- Goal test: All regions are uniquely colored relative to their neighbor

b.

- Initial state: 3 foot monkey in a room with bananas suspended 8 feet above it

- Successor: The monkey moves the crate to be able to reach a banana

- Path cost: The amount of actions the monkey takes

- Goal test: The monkey has all the bananas

c.

- Initial state: The inputs

- Successor: Processing the inputs

- Path cost: The time it takes to process the inputs

- Goal test: Identifying the inputs to be illegal or not

d.

- Initial state: Having the various jugs and water faucet

- Successor: Filling or emptying the jugs

- Path cost: Total number of actions

- Goal test: Measuring one gallon

**3.8:**

a. An optimal solution would have to explore the entire space in order to find out if an arbitrary negative cost action can be done to find the least expensive path.

b. By including a negative constant, the algorithm will know that the best case scenario is said negative constant, and thus search for these best-case scenarios.

c. The agent will infinitely loop because it will detect that this is the optimal solution.

d. In a real-world scenario, the states such as the given example will change. The cost of actions will retroactively change, with some actions becoming more expensive (i.e. the scenic route becoming more expensive as the person gets sick of seeing the scenery), and vice versa.

e. An example of a real domain looping would be a person going to class every day.