Full Name:______________________________________

CS 4300, Fall 2018

Written Exam I

Sep 27, 2018

Scenario 1

The intersection of University Avenue and St. George Boulevard handles a lot of traffic, both vehicles and pedestrians. Traffic lights, including crosswalk lights, control the flow of traffic. There are crosswalks for all four crossing directions. Our agent is located on the south-east corner of the intersection and needs to reach the north-west corner. The agent is a physical machine, about 3 feet tall with a doom shaped top. The agent has audio and visual sensors, and motion actuators that allow the agent to move forward and rotate to change direction. The agent carries vital information that must reach the north-west corner to save the inhabitants of Washington County from a life of oppression.

Problem 1. (2 points):

Describe the performance measure you would use for this environment. Explain each term of the formula.

Problem 2. (2 points):

What are the important percepts a software developer would want to derive from the sensors?

Problem 3. (2 points):
What are the important actions a software developer would want to use from the actuators?

Problem 4. (2 points):
Would you classify this environment as Fully or Partially Observable? Why?

Problem 5. (2 points):
Would you classify this environment as Episodic or Sequential? Why?

Problem 6. (2 points):
Would you classify this environment as Deterministic or Stochastic? Why?

Problem 7. (2 points):
Would you classify this environment as Static or Dynamic? Why?

Problem 8. (2 points):

Would you classify this environment as Single or Multi Agent? Why?

Problem 9. (2 points):

Would you classify this environment as Discrete or Continuous? Why?

Problem 10. (2 points):

Would you classify this environment as Known or Unknown? Why?

Problem 11. (2 points):

What type of agent implementation (e.g. Reflex, Model, Goal, Utility, etc.) would you choose to create an agent that would perform well in this environment? Why?

Scenario 2

The puzzle shown is a variation of slider puzzles. Legal moves are made by sliding a piece into an unoccupied area. Each slide is a separate move. In this puzzle, the pieces have a variety of shapes. The goal is to move the big red piece to the bottom of the box next to the red line on the border. Our agent is a digital agent. It is presented with a digital copy of the problem in a suitable format and returns a sequence of actions in a suitable format.

For this set of problems, consider implementing the Supercompo agent using classic search. If you don't have sufficient information to answer any question, make your best guess, and state why you feel it is inaccurate.

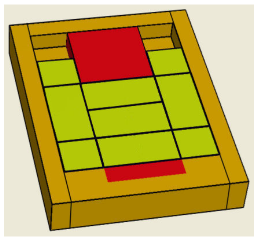
Figure 1: Supercompo Puzzle

Problem 12. (2 points):

What is the maximum value of b, the branching factor? Be sure to consider advanced states of the puzzle, not just the start state.

Problem 13. (2 points):

What is the value of m, the maximum tree depth?

Problem 14. (2 points):

What is the value of d, the depth of the shallowest goal?

Problem 15. (2 points):

Would you use tree or graph search for this problem? Why?

Problem 16. (2 points):

Which frontier variety (e.g. DFS, DL, IDS, BFS, UC, A*, etc.) would you use? Why?

Problem 17. (2 points):

Are your choices for the previous two questions consistent? Why?

Scenario 3

A particular search problem has no heuristic available, but needs to be solved as efficiently as possible. Analysis shows that the search tree for the problem is finite with a maximum depth of 10. Each state has up to 3 legal actions. The goal states are only found at the maximum depth. Each problem instance has many goal states; approximately 50% of the maximum depth states are goals, spread across the maximum depth. This search needs to take place on a micro-server, quickly finding its solutions to keep lag times for users minimum. Each solution needs to be found in less than 0.5 seconds. Because of the nature of the problem, each node takes approximately 0.001 seconds to expand. Note: $3^{10} = 59049$.

Problem 18. (8 points):

Write the Big-O limits on the time and space complexity for BFS and DSF on this problem. Give numbers and the formulas used to calculate them.

Problem 19. (2 points):

Which classic tree search strategy (not limited to BFS and DFS) would you use for this problem? Why?

Scenario 4

Description copied verbatim from Wikipedia.

The canonical Kakuro puzzle is played in a grid of filled and barred cells, ”black” and ”white” respectively. Puzzles are usually 16×16 in size, although these dimensions can vary widely. Apart from the top row and leftmost column which are entirely black, the grid is divided into ”entries”—lines of white cells—by the black cells. The black cells contain a diagonal slash from upper-left to lower-right and a number in one or both halves, such that each horizontal entry has a number in the black half-cell to its immediate left and each vertical entry has a number in the black half-cell immediately above it. These numbers, borrowing crossword terminology, are commonly called ”clues”.

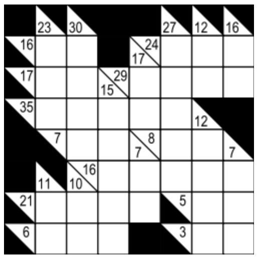
Figure 2: Kakuro Puzzle

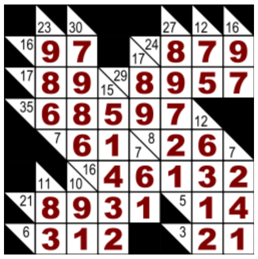
Figure 3: Kakuro Solution

The objective of the puzzle is to insert a digit from 1 to 9 inclusive into each white cell such that the sum of the numbers in each entry matches the clue associated with it and that no digit is duplicated in any entry. It is that lack of duplication that makes creating Kakuro puzzles with unique solutions possible, and which means solving a Kakuro puzzle involves investigating combinations more, compared to Sudoku in which the focus is on permutations. There is an unwritten rule for making Kakuro puzzles that each clue must have at least two numbers that add up to it, since including only one number is mathematically trivial when solving Kakuro puzzles.

For this set of problems, consider implementing the Kakuro agent using local search. If you don’t have sufficient information to answer any question, make your best guess, and state why you feel it is inaccurate.

Problem 20. (2 points):

Describe a state of this problem.

Problem 21. (2 points):

Assume gradient descent is to be used. Describe a suitable utility function.

Problem 22. (2 points):
Assume gradient descent is to be used. Describe the neighbors of a state.

Problem 23. (2 points):
Assume gradient descent is to be used. Would you expect to need random restart? Why?

Problem 24. (2 points):
Assume the genetic algorithm is to be used. Describe a suitable genome.

Problem 25. (2 points):
Assume the genetic algorithm is to be used. Describe a suitable mutation operation.