CS471

Puzzle System

Juan Solis Melgar

Completed

**==========================================================**

**Check List (This must be checked off and included in your Cover Page):**

1. ** Did you follow all the requirements in implementing the system?**
2. **Did you create one Word file of your report with the cover page and section headers as specified?**



1. **Did you answer all questions per section?**



===========================================================

**Section 1: Analyzing the Problem Space**

1. How many unique states (include all possible whether reachable/legal or not)? = 60
2. Show the equation on how you came up with this number.

Chart

Description automatically generated with medium confidence

1. List all possible disk move actions/operators (what goes where):

Let’s assume the Large disk = L, Medium disk = M, Small disk = S

Initial state: [S,M,L] [ , , ,] [ , , ,]

1. [ ,M,L] [ , , ,] [ , ,S]
2. [ , ,L] [ , , M] [ , ,S]
3. [ , ,L] [ ,S,M] [ , , ,]
4. [ , , ,] [ ,S,M] [ , , L]
5. [ , ,S] [ , ,M] [ , ,L]
6. [ , ,S] [ , , ,] [ ,M,L]
7. [ , , ,] [ , , ,] [S,M,L]
8. Therefore, the Branching factor: <= 7 because there are only 7 legal operators.

**Section 2: Drawing the Problem Space**

1. Draw all states that are reachable/legal, and draw all possible arrows between them to create a graph. No node shall be duplicated (i.e. one node per state).
2. Label the arrows with moves (you may use a “legend” to make it easy to label links).
3. Diagram

   Description automatically generatedMark the initial and goal states.

**Section 3: Designing the Evaluation Function**

**Assume that g = number of disk moves so far and**

**h = estimate of how many more moves a given state**

**f = g+h is the goodness**

1. Give the equation for the estimate function **h** (i.e. how do you guess the # of moves?)

**h** = (2^**n**) – 1 – **g** where **n** is the number of disks.

1. Give an example **h** value and **g** value for **3** of the states in the above Problem Space.

**h** = (2^3) – 1 – 3

**h** = 4

1. Defend your decision for the **h** function:
2. Is this a good estimate of how many more moves are required? Yes. If the player plays perfect it will take a total of 2^n – 1 moves to get to the goal state. Thus, **h** will never be higher than the moves left to reach the goal state.
3. Is this h admissible? Explain. Yes. Since we are always deducting h by the amount g we will never overestimate. If the player plays perfect it will require 7 steps to reach the goal state. From the above equation the initial value for h is equal to 7, which means even if the player plays perfect h will always be admissible.

**Section 4: Implementation (Source Code is submitted separately)**

1. Must have functions dedicated to do the following. Write the name of the function next to each:

Note: I used a class to keep track of each peg

* 1. Basic framework of puzzle Name: state File: main.cpp
  2. Generating the new states Name: generateAllStates File: main.cpp
  3. Choosing the next state to expand Name: checkState File: main.cpp

1. What data structure (type and name) did you use for each node/state? I used a class named state to hold all the information for each state.

Text

Description automatically generated

* 1. Give a picture of it with examples values.A screenshot of a computer

     Description automatically generated with medium confidence

1. What data structure (type and name) did you use to store all the Frontier nodes/states?

I used a vector of type state named frontier.

**Section 5: Testing and Results**

1. Test your program thoroughly and make sure the output matches your expectations.
2. Include here the test results (screen snapshots/recorded script compiled with g++).

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

1. Your analysis of the test results:
2. Did it work as expected? If not, explain. Yes, it did work as expected.
3. How many disk moves did it take to reach the goal? 11
4. Is that optimal? Or a person can do with fewer moves? 11 moves are not optimal because a person can solve the Hanoi puzzle with only 7 moves.

**Section 6: Ideas for Adding Machine Learning**

1. **Give one way to incorporate machine learning into playing puzzles in general**

* What part of the program will the puzzle program update to improve itself? One way to make the program more optimal is for it to learn which states are not legal. By running the program multiple times it will learn which states to avoid in order to solve the problem with less steps.

1. **Then answer the following questions for your own project program:**

* How and when would this (from #1) learning happen?

(Being advised by a human as it plays?

Automatically by observing an outcome?) Be specific.

The program will achieve an optimal solution automatically by running itself multiple times until it finds the correct path. Even if there is more than 3 disks it will find which paths to avoid until it finds the most optimal path. We can implement a min-max tree with a limited search to look ahead and cut off a path that will lead to a illegal move.