CS 471

Final Project: Puzzle System (Towers of Hanoi)

Fernando Alvarado

State of System: 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?

3. \_✓\_\_ 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.

I separated the equation into three different parts because of the three different kinds of states the puzzle can be in:

The first kind of state is when all three disks are on one peg. I find the permutation of 3P3 to find how many different ways there are to order the three disks onto one peg. Then multiply that by three because there are three pegs.

(3P3) \* 3

The second kind of state is when there are two disks on one peg and one disk on some other peg. I do the same thing as last time where I find the permutations for a single peg and multiply that by three, (3P2) \* 3. But now we have to factor in that there are two different pegs that the extra disk can be on so we multiply the equation by two

(3P2) \* 3 \* 2

The last kind of state is when each disk is on its own peg, and again we find the permutations but this time we can’t multiply by three because all other pegs are taken in this situation, but we still do multiply by 2 because the disks on the other two pegs can switch for each permutation on the first peg.

(3P1) \* 2

Then we add all of the parts together:

(3P3) \* 3 + (3P2) \* 3 \* 2 + (3P1) \* 2 = 60

(nPn) \* n + (nP(n-1)) \* n \* (n-1) + (nP(n-2)) \* (n-1)

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

S = small disk M = medium disk L = large disk x = current peg #

S goes to P(x+1)%3

S goes to P(x+2)%3

M goes to P(x+1)%3

M goes to P(x+2)%3

L goes to P(x+1)%3

L goes to P(x+2)%3

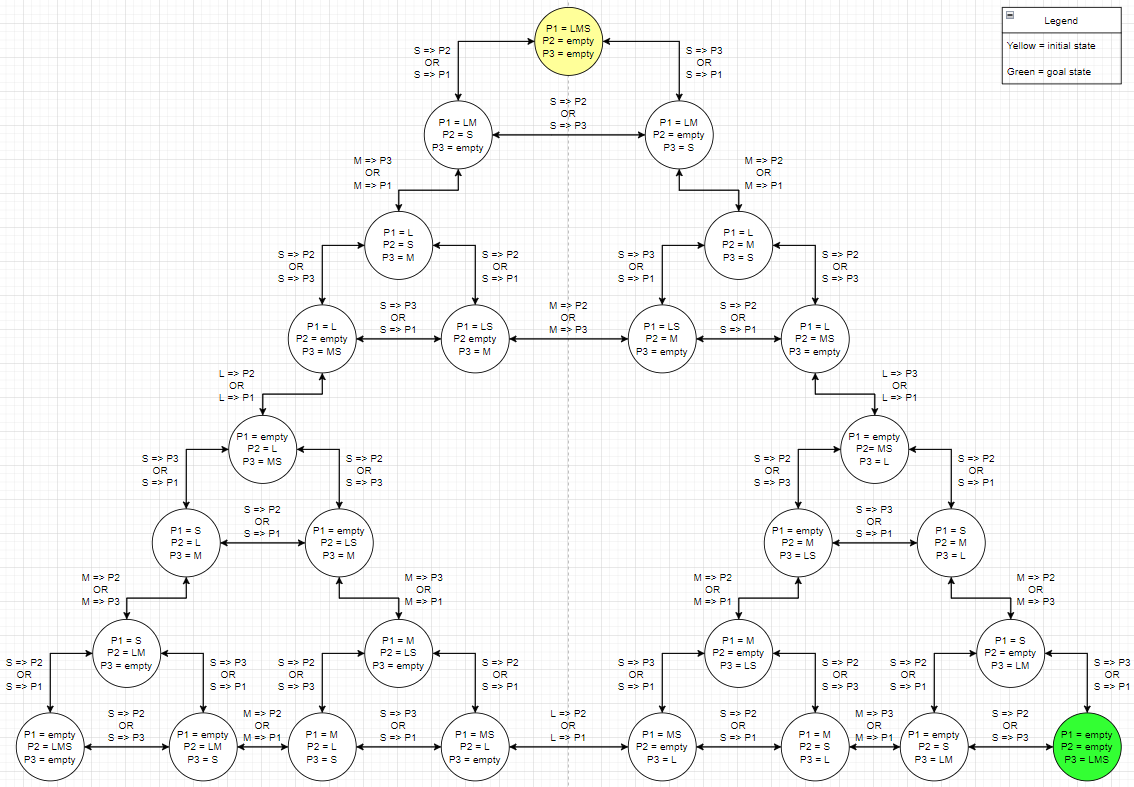
4. Therefore, the Branching factor: <= 6

**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. Mark 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 = 3 – number of disks on peg 3

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

P1 = L, P2 = M, P3 = S

g = 2

h = 2

P1 = S, P2 = empty, P3 = LM

g = 6

h = 1

P1 = empty, P2 = LMS, P3 = empty

g = 7

h = 3

3. Defend your decision for the **h** function:

a. Is this a good estimate of how many more moves are required?

This is a good estimate of how many more moves are required because you will need to make at least 3 moves in order to get all three disks onto the goal peg if none are already on there 2 if one is already on and so on.

b. Is this h admissible? Explain.

This h is admissible because since we are calculating based off of how many disks are already on the goal peg we will never overestimate.

**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:

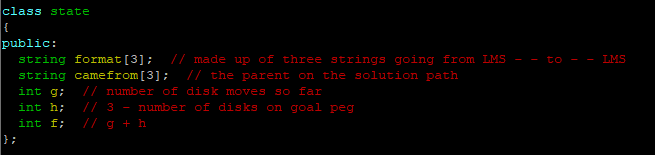
a. Basic framework of puzzle Name: framework File: main.cpp

b. Generating the new states Name: generateAll File: main.cpp

c. Choosing the next state to expand Name: bestofFrontier File: main.cpp

2. What data structure (type and name) did you use for each node/state?

a. Give a picture of it with examples values.



example values:

format = [“L” , “MS”, “”]

camefrom = [“L”, “M” , “S”]

g = 3

h = 2

f = 5

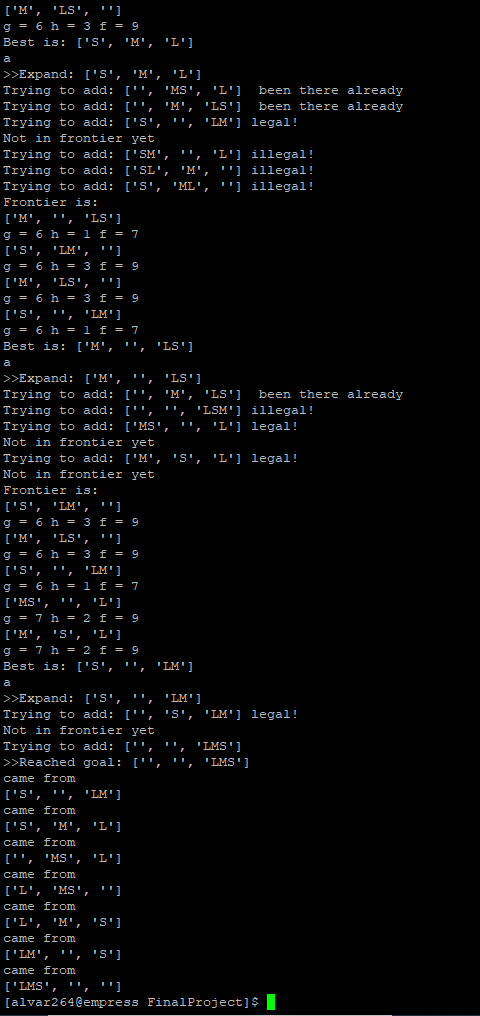
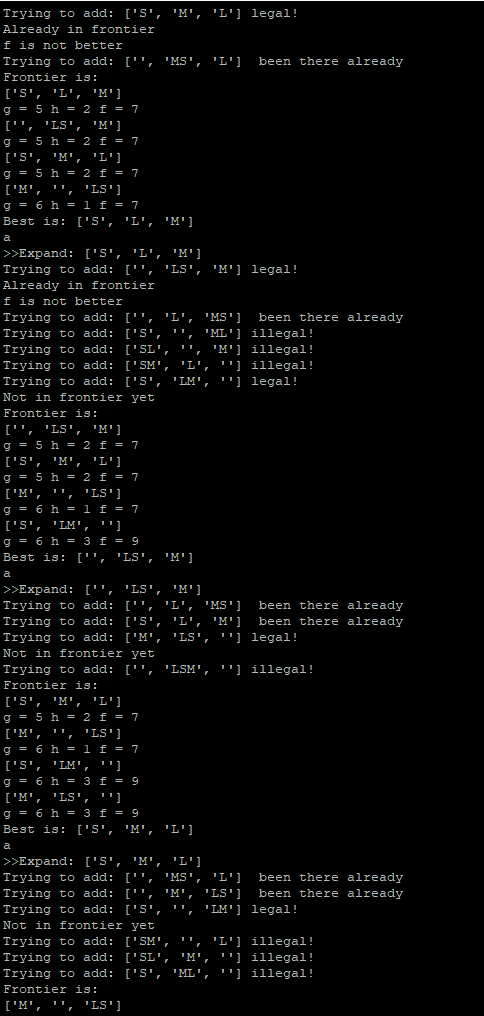
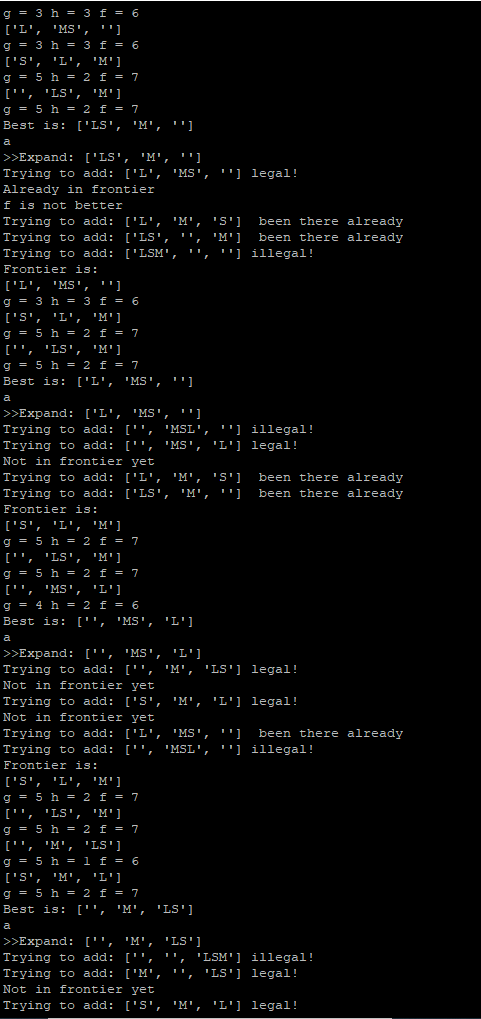
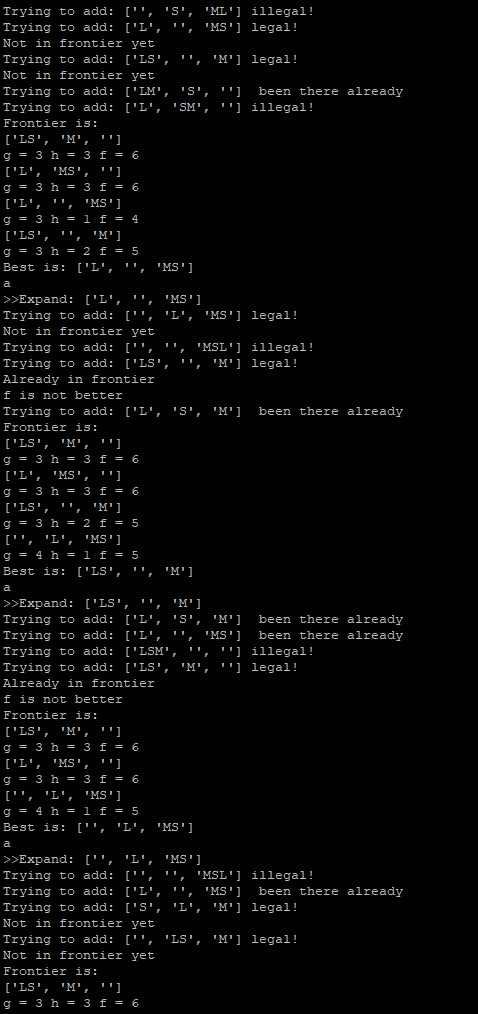
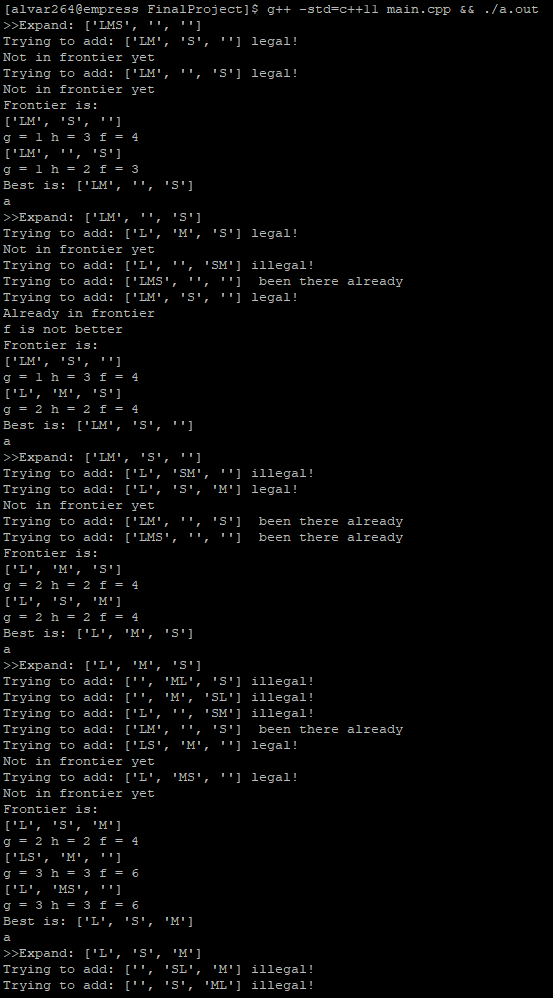
3. What data structure (type and name) did you use to store all the Frontier nodes/states?

I used a vector of states and called it 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++).



3. Your analysis of the test results:

a. Did it work as expected? If not, explain.

It does work just not as I expected because in the beginning I had somewhat forgotten how a program like this works and I just assumed it would take the best route the whole way through and ignore all other routes but instead it analyzed each route, and ended up coming back to the best route and finishing on the best route.

b. How many disk moves did it take to reach the goal?

7 moves

c. Is that optimal? Or a person can do with fewer moves?

that is the optimal amount of 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?

I think the program could change the way it calculates h in order to get an even more accurate estimate without ever overstimating

**2.** **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.

This learning would happen automatically after completing the puzzle for the first time. I think this would be the best time to do it because the program can go back and see how many times it strayed away from the best path and adjust according to the previous h’s