Project Extra Credit2

Slide Puzzle Solver

Due in Blackboard Dec. 14, 2012 11:30 pm

Each student should submit a tar.gz file containing all sourcecode including Makefile, solver.cpp and all necessary files to Blackboard. No hard-copy required.

This extra-credit assignment will replace your 3rd lowest programming assignment or quiz grade, whichever is lower, as long as this grade is higher. The lowest programming and quizzes are already dropped. Thus, this can only help your grade.

In this project you’ll be making a generic slide puzzle solver. See the Wikipedia article for details of the slide puzzle.

The goal of this project is to create an executable which will print out a sequence of moves to transform one game state into another. A move is one of the following characters: U, D, L, R. These correspond to moving the blank square. Thus, most game states have 4 possible moves, some 3, and others 2.

Input states into the program should be in linear string form. Thus ‘123_’, corresponds to the state:

```
1
2
3
```

To solve a game, input the input state, and the desired goal state. Any size board should be allowed up to a 6x6, and each character in the state should be unique. Any characters are allowed. The example above is a 2x2. Here is an example:

```
./solver _213 123_
```

Can be solved with 2 moves: D,R.

Hints:

To find a set of moves between two states, imagine a directed graph, with board states as vertices. An edge between states represents a move. For example performing an ‘L’ transition can be undone by doing another ‘R’ move back to the original state.

A set of moves to get from one board state to another is therefore a path between the states. You can use BFS or DFS traversal of this graph to find some path between the
states. Since we’re looking for the shortest path between two states the BFS traversal will find the shortest solution first. Be sure to never visit the same state twice.

Note that two random board states may not have a path between them.

Your program is required to output the shortest set of moves between the given states. The example below provides more information than is necessary.

Your completed project should contain a Makefile. When ‘make’ is run in the folder a ‘solver’ executable should be made.

You’ll be graded on your code and the use of learned data structures, and the Makefile.

Your grade will consist of the following:

1. 20% - Function headers for public functions, files, classes, comments, code indentation.
2. 70% - Proper operation of solver to solve length 5 solutions or less.
3. 10% - Ability to solve 3x3 problems with solutions of 20 or less.

**Extra Credit:**

+5 points: Be able to solve 4x4 problems with solutions of 50 or less, in less than 1 minute. Does not need to be optimal. A* will be needed. (Look up online, ask me, or look in AI book)

**Example States & Solutions:**

Note that other solutions may exist with the same number, or more, moves.
31_2 -> 123_ with 3 moves, URD.

312_ -> 123_ with 4 moves, LURD.

1524837_6 -> 12345678_ with 5 moves, UURDD.

15248376_ -> 12345678_ with 6 moves, LUURDD.

127356a490e8cdb_ -> 1234567890abcde_ with 8 moves, LUUURDDD.

1854_2763 -> 12345678_ with 10 moves, URDDLURDD.

4187_5362 -> 12345678_ with 20 moves, DLUURDDLUURDDLUURDD.

91a7c2530e46db8_ -> 1234567890abcde_ with 36 moves,
LLULURRURDDLURDDLURULURURDDLUURRRUURDDDD.

a917c2530e46db8_ -> 1234567890abcde_ with 48 moves,
LLULURRDRDLLLLUURUULDRURDDDLLDLUURRDLDRUULDDRUURDD.