9/29/2016 8:39 AM

CSCE 420
Programing Assignment #2
due: Fri, Oct 14, 2016 (by midnight)

Objective

The overall goal of this assignment is to implement **A* search** and use it to solve a classic AI search problem: the "Blocksworld". This problem involves stacking blocks into a target configuration. You are to write a program to solve random instances of this problem using your own implementation of A* search. The focus of the project, however, is to use your intuition to develop a *heuristic* that will make solving various instances of this problem efficient.

This project will build on code you wrote for programming assignments 1 and 2. Specifically, you will use a queue-based search function, where the *priority_queue* is kept sorted based on f(n)=g(n)+h(n). g(n) is just the path length from the root to the current node. **The main focus will be on developing a heuristic function**, h(n), for this domain that estimates the distance to the goal (number of moves to solve the problem) and testing how it affects the efficiency of the search.

Description of the problem

This task involves stacking blocks. The blocks are labeled by letters. Here is an example random start state and goal state for a problem with 5 blocks on 4 stacks.

```
example Start state:   example Goal state:
                           B
                           C
     A     E               D
  D  C     B            A  E
  ----------            ----------
  1  2  3  4            1  2  3  4

    State1                State2
```

The state can be represented as a list of lists, or an array. However, you should make your code flexible so that you can test it on different numbers of blocks and stacks (as parameters). The initial and goal states will be given as input files on the command line. The format of the input file is: a space-separated list of the blocks in each stack, one per line. For example, the input files for the 2 states above look like:

```
State1.txt:
1 D
2 C A
3
4 B E

State2.txt:
1 A
2 E D C B
3
4
```

Note that the first item on each line is the number of the stack. (stacks 3 and 4 in State2.txt are empty) You can assume the files will have the same number of blocks and stacks. You can assume there will be at most 20 stacks and 26 blocks.

The operator in this problem can pick up only the top block on each stack and move it to the top of any other stack. For example, here are the 9 successors of State1:

```
   D                                           D
   A     E             A     E             A     E
   C     B             C  D  B             C     B
----------          ----------          ----------
1  2  3  4          1  2  3  4          1  2  3  4

                                               A
A        E                   E                 E
D  C     B          D  C  A  B          D  C     B
----------          ----------          ----------
1  2  3  4          1  2  3  4          1  2  3  4

                       E
E  A                   A                   A
D  C     B          D  C     B          D  C  E  B
----------          ----------          ----------
1  2  3  4          1  2  3  4          1  2  3  4
```

This problem becomes harder as the number of blocks scales up, and cannot be solved effectively with BFS (though you can try it for small numbers of blocks). You will need A* search coupled with a heuristic to find solutions efficiently. The default heuristic (call it h_0) can be taken to be the "number of blocks out of place". Your goal is to define a better heuristic(s) that will enable your algorithm to find solutions faster (with fewer goal tests), and to be able to solve larger problems (with more blocks).

You should evaluate your heuristic with several different parameter settings which control the difficulty of the problems. For example, start with 5 blocks and 3 stacks, then try 6-10 blocks to see how scalable your performance is. Then try expanding the number of stacks from 3 to 7, to see if this makes the problem easier or harder. See if you can solve "hard" problems, like with 10 blocks on 6 stacks, as shown below (the solution has about 20 steps):

```
                        E              J
                        D              I
      F     D  H        C              H
      I     G  B        B              G
C     A     E  J        A              F
----------------        ----------------
1  2  3  4  5  6        1  2  3  4  5  6
  State5.txt               State6.txt
```

Your heuristic does not have to be provably admissible, though the closer to admissible it is, the more efficient it will be.

Implementation

You should use C++ or Java.

You will need to write a new Node class for representing states in the Blockworld, and a successor() function for generating moves.

Make your implementation flexible so it can solve problems with an arbitrary number of blocks and an arbitrary number of stacks (e.g. provided as command-line arguments).

As before, you will have to have a method for checking for *visited states*.

What to Turn in

- You will submit your code for testing using the web-based CSCE *turnin* facility, which is described here: https://wiki.cse.tamu.edu/index.php/Turning_in_Assignments_on_CSNet
- You should include a Word document with instructions on how to compile and run your program,
- Include a transcript that shows **example program traces** (transcripts) for your A* search, including 2 problems with solutions requiring ~5 steps, 2 problems with path length ~10 steps, 2 with length ~15, and 2 with length ~20 (if you can).
- Include a description of how your heuristic works.

Example Transcript

```
490 sun.cs.tamu.edu> blocksworld State1.txt State2.txt
initial state
1 | D
2 | C A
3 |
4 | B E
goal state
1 | A
2 | E D C B
3 |
4 |
iter=0, g(n)=0, h(n)=5.0 g+h=5.0, frontier_size=0
1 | D
2 | C A
3 |
4 | B E
iter=1, g(n)=1, h(n)=5.0 g+h=6.0, frontier_size=8
1 |
2 | C A D
3 |
4 | B E
...
```

```
...
iter=101, g(n)=3, h(n)=4.0 g+h=7.0, frontier_size=814
1 | A
2 | C
3 | D
4 | B E
iter=102, g(n)=7, h(n)=0.0 g+h=7.0, frontier_size=825
solution found!
total iterations=103
max queue size=826
path length=8
step 1
1 | D
2 | C A
3 |
4 | B E
step 2
1 |
2 | C A
3 | D
4 | B E
step 3
1 | A
2 | C
3 | D
4 | B E
step 4
1 | A C
2 |
3 | D
4 | B E
step 5
1 | A C
2 | E
3 | D
4 | B
step 6
1 | A C
2 | E D
3 |
4 | B
step 7
1 | A
2 | E D C
3 |
4 | B
step 8
1 | A
2 | E D C B
3 |
4 |
```