FUNDAMENTALS OF ARTIFICIAL INTELLIGENCE - CS161

Programming Assignment 7 - Due 11:59, Tuesday, February 26

DO NOT CONSULT ANY OUTSIDE REFERENCES OR SOURCES FOR THIS ASSIGNMENT

Your assignment is to use the ideas of the General Problem Solver (GPS) to write a LISP program to solve the Towers of Hanoi problem. Your program must generate a solution from ANY legal initial state to ANY legal goal state with a minimum number of moves, and work for any number of discs.

The Towers of Hanoi problem consists of three pegs labelled A, B, and C, and an arbitrary number of discs, numbered 1, 2, 3, etc. from smallest to largest. A legal state is one in which no larger disc is on top of a smaller disc. A legal move is to move the top disc on any peg to any other peg as long as it is not placed on top of a smaller disc.

Your top-level function should take two arguments, the initial state and the goal state. You may represent a state any way that you wish, but your choice of a state representation will have a large impact on the ease of programming. THE KEY TO THIS PROBLEM IS CHOOSING A GOOD STATE REPRESENTATION. The value returned should be a simple list of move instructions of the form (disc source destination), which means move the numbered disc from the source peg to the destination peg. For example, the first initial and goal states shown below should return the solution: ((2 C B) (1 A B) (4 A C) (1 B C) (2 B A)).

You should rigorously test your program. Because picking a good representation is the key to this problem, the form of parameters that your function should take are unspecified. Pick a good state representation and document it. Because we are not specifying the input, you should also define a set of functions, which take no arguments, which run your program on each of the following inputs below and on the back. The names of the functions that should compute the result are given with each input. The implementation of the test functions should merely call your real function with an input representing the start and goal states depicted below and on the back.

Test Function	Start	Goal
test1	1 4 3 2 A B C	1 2 3 4 A B C
test2	1 2 3 A B C	3 2 1 A B C
test3	1 2 3 A B C	1 2 3 A B C
test4	1 2 5 4 3 A B C	2 1 3 4 5 A B C
test5	3 2 1 6 5 4 A B C	1 2 3 6 4 5 A B C
test6	1 2 3 4 5 6 A B C	2 1 4 3 6 5 A B C
test7	1 4 2 6 5 3 A B C	1 3 4 6 2 5 A B C
test8	1	1

```
                2           2
                3           3
                4           4
                5           5
                6           6
                7           7
                A B C   A B C

test9           1       1
                2 5     2 5
                3 6 8   3 6 8
                4 7 9   4 7 9
                A B C   A B C

test10          A B C   A B C

test11          1       1
                2 5     3   2
                3 6     5   4
                4 7     7   6
                A B C   A B C

test12          1 4       2 1
                2 5 6   3 4 5
                3 8 7   7 6 8
                A B C   A B C
```