Homework 2

CS 4710

Due: 2/27 at 11:59PM EST

Problem 1

Recall the 8-puzzle from our class notes on state search.

Write the following python functions:

(1) A generator function that, from scratch, can generate a random 8-puzzle state. (Note: you must determine how to represent the state. This function is so that you can test your own program.)

(2) A test function that determines if an 8-puzzle state is the goal state (defined in the notes).

(3) A solver function that accepts an arbitrary 8-puzzle state, and (a) if solvable, returns a finite chain of transitions leading to the goal, and (b) if not solvable, returns the phrase "unsolvable". Your solver function should accept a single parameter (an array) and be called *solve* – the 8-puzzle's first row, second row, and third row should be the three elements of this input array (precise spec below). This allows TA's to easily test your function despite what your underlying state representations are.

You will receive input in JSON format as a string. As an example:

```
[[1,2,3],[0,4,5],[6,8,7]]
```

When grading your program, your TA's (and I) will produce JSON structures as listed above and feed them directly into your solver function (produced in 3 above). Be sure it handles this format.

Show the following:

(4) Write your search algorithm formally (representing the states, the state transitions, and so on) and argue that your representation is adequate.

(5) Prove whether your algorithm is or isn't **correct**.

(6) Prove whether your algorithm is or isn't **complete**.

(7) Prove your algorithm solves the 8-puzzle in the minimum amount of steps (note: this requires having written such an algorithm).

Problem 2

(1) Write a python function called *construct_tree* that accepts as input any list of well-formed statements of propositional logic (PC). For malformed strings, it should return the string "INVALID-INPUT". Let "|" represent the $\vee$, "&" represent $\wedge$, "!" represent $\neg$, ">" represent $\rightarrow$.

construct_tree should should take the input set and return, using a set of finished PL trees, whether or not the input set forms a valid argument. Be sure your function also prints the tree in some clear fashion.

Here's some example inputs:

```
["!A | B", "A", "B"]
["A & B", "B", "B > C", "!C"]
```

(2) Prove your algorithm is **correct**.

Problem 3

Suppose you have a search tree over an arbitrary search space. Now suppose we wish to find the minimum path (where each edge is counted as 1) between two nodes n and m. Prove that BFS finds the shortest path.