B551 Assignment 0: N-queens and Python

Spring 2015
Due: Tuesday September 6, 11:59PM Eastern (New York) time
(You may submit up to 48 hours late for a 10% penalty.)

This assignment will give you practice with posing AI problems as search and an opportunity to dust off your coding skills. Because it's important for everyone to get up-to-speed on Python, for this particular assignment you must work individually. Future assignments will allow you to work in groups. Please read the instructions below carefully; we cannot accept any submissions that do follow the instructions given here. Most importantly: please **start early,** and ask questions on Piazza or in office hours.

If you don't know Python, never fear – this is your chance to learn! But you'll have to spend some significant amount of out of class time to do it. We recommend the Python CodeAcademy website, `http://www.codeacademy.com/learn/python`, and Google's Python Class, `https://developers.google.com/edu/python/`. There are also a wide variety of tutorials available online. The instructors are also happy to help during office hours.

Academic integrity. You may discuss the assignment with other people at a high level, e.g. discussing general strategies to solve the problem, talking about Python syntax and features, etc. You may also consult printed and/or online references, including books, tutorials, etc., but you must cite these materials (e.g. in source code comments). However, the work and code that you submit must be your own work, which you personally designed and wrote. You may not share written answers or code with any other students, nor may you possess code written by another student, either in whole or in part, regardless of format.

Introduction

We discussed the 8-queens problem before, where the goal is to place 8 queens on an 8x8 chessboard such that no two queens share the same row, column, or diagonal. This is a specific case of the N-queens problem, which involves place N queens on an NxN chessboard. Not all N's have solutions: there's a (trivial!) solution for N=1, no solution for 2 and 3, but there are solutions for N=4, for example.

To get you started, we've already written a solver for a related, simpler problem – N-rooks. N-rooks is like N-queens, except that the goal is to place N rooks so that no two of them are in the same row or column. This is a lot easier because there are many more possible solutions, since we don't need to worry about diagonals. Our code is available via Canvas. You can adjust the first line of code to set the value of N you're interested in, and then it will try to find a solution and print out the first one it finds. The problem is that it's really slow. You can easily test that it works for N=1 and N=2, but larger values of N seem to take a really long time.

What to do

Let's fix the implementation and mathematical abstraction to make it much faster, and then use it to solve N-queens.

1. Spend some time familiarizing yourself with the code. Write down the precise abstraction that the program is using. In other words, what is the set of valid states, the successor function, the cost function, the goal state definition, and the initial state?
2. Is the algorithm using BFS or DFS? Switch to the opposite, and explain exactly how to modify the code to do this. What happens now for N=4 and N=8?
3. The successor function in the code is defined in a very simplistic way, including generating states that have N+1 rooks on them, and allowing "moves" that involve not adding a rook at all. Create a new successors() function called successors2() that fixes these two problems. Now does the choice of BFS or DFS matter? Why or why not?

4. Even with the modifications so far, N=8 is still very slow. Recall from Section 2.8 that we could define our state space and successor functions in a smarter way, to reduce the search space to be more tractable. Describe your new abstraction, and then modify the code to implement it with a new successors function called successors3(). Feel free to make other code improvements as well. What is the largest value of N your new version can run on within about 1 minute?

   Tip: In Linux, you can use the `timeout` command to kill a program after a specified time period:

   ```
   timeout 1m python nrooks.py
   ```

5. Now, modify your code so that it solves and displays solutions for both the N-rooks and N-queens problem, using the enhancements above. What is the largest value of N that your new version can solve within 1 minute?

What to turn in

Turn in two files via Canvas: (1) a PDF or text document that answers questions 1, 2, 3, 4, and 5, and (2) your final Python source code. Make sure that your code runs correct on the CS Linux servers, `burrow.soic.indiana.edu`, since this is where we'll test your code. *We will deduct points if we have to modify your code in order to make it run on our system.*