B551 Assignment 0: N-queens and Python

Fall 2017
Due: Sunday September 10, 11:59:59PM 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 not 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. Students in past years have recommended 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. If you're already proficient in one particular language, you might look for tutorials with names like "How to code in Python: A guide for C# programmers," "Python for Java programmers," etc. Feel free to share any particularly good or bad resources on Slack and/or Piazza. The instructors are also happy to help during office hours.

Academic integrity. We take academic integrity very seriously and, to maintain fairness to all students in the class and integrity of our grading system, we will prosecute any academic integrity violations that we discover. *Before beginning this assignment, make sure you are familiar with the Academic Integrity policy of the course, as stated in the Syllabus, and ask us about any doubts or questions you may have.* To briefly summarize, 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 in class, 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 placing 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 specify a 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.

You can run it like this (on the SOIC Linux machines):

```
./nrooks.py N
```

where N is the number of rooks you're interested in. Note that you may first have to change the Unix permissions on the nrooks.py file to tell the operating system that it is a program, like this:

```
chmod u+x nrooks.py
```

What to do

1. Spend some time familiarizing yourself with the code. Write down the precise abstraction that the program is using and include it in your report. 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. The current code is far from efficient. Recall from class that there are usually many ways to define a state space and successor function, and some are more practical than others (e.g. some have much larger search spaces). Create a more efficient successor function called successors2(). Hint: avoid generating states that have N+1 rooks on them or allowing "moves" that involve not adding a rook at all.

3. You've probably noticed that the code given is implemented by depth first search (DFS). Modify the code to switch to BFS instead.

4. Run your program and measure the time required for both BFS and DFS for various values of N (e.g. from 2 to 6). (On a Linux machine, you can use the `time` command.) Draw a figure or create a table showing how running time varies with N. Explain the behavior you notice – why are the two similar or different?

5. Now, create a new program, `a0.py`, with several additional features, including the ability to solve both the N-rooks and N-queens problems, and the ability to solve these problems when when exactly one position on the board is unavailable (meaning that you cannot place a piece there). Your program must accept four arguments. The first one is either `nrook` or `nqueen`, which signifies the problem type. The second is the number of rooks or queens (i.e., N). The last two arguments represent the coordinate of the unavailable position. For example, `2 3` means the square at the second row and the third column is unavailable. Assuming a coordinate system where (1,1) is at the top-left of the board. Taking the 7-rooks problem with position (1, 1) unavailable as an example, the format and one possible result could be:

   ```
   [<>djcran@tank ~]$ python a0.py nrook 7 1 1
   X R _ _ _ _ _
   R _ _ _ _ _ _
   _ _ R _ _ _ _
   _ _ _ R _ _ _
   _ _ _ _ R _ _
   _ _ _ _ _ R _
   _ _ _ _ _ _ R
   ```

   where `R` indicates the position of a rook, underscore marks an empty square, and `X` shows the unavailable position. Or for the 8-queens problem with (1, 2) unavailable, one possible result could be:

   ```
   [<>djcran@tank ~]$ python a0.py nqueen 8 1 2
   _ X _ _ Q _ _ _
   _ _ _ _ _ _ Q _
   _ Q _ _ _ _ _ _
   _ _ _ _ _ Q _ _
   _ _ Q _ _ _ _ _
   Q _ _ _ _ _ _ _
   _ _ _ Q _ _ _ _
   _ _ _ _ _ _ _ Q
   ```

   where the `Q`'s indicate the positions of queens on the board. As a special case, the coordinate `0 0` indicates that there are no unavailable squares. Please print only the solution in exactly the above format **and nothing else**. The output format is important because we will use an auto-grading script to test and grade your code.

Hints and tips

Output format instruction. In the assignments in this class, we will usually give you specifications on the format of the input and output to your program, and you must be very careful to satisfy these specifications. This is for two reasons: (1) we typically test your code with a semi-automatic grading script that expects input and output in a particular format, and (2) an important part of becoming a better programmer is to learn how to work within a client's precise specifications. Make sure your code exactly satisfies the input and output specifications given above.

Testing your code. It's very important that you test your code on the CS Linux servers, `burrow.soic.indiana.edu` (aka silo), before you submit it. Because of differences in Python versions, operating systems, etc., your code may work beautifully on your home computer but fail to run on the CS Linux servers, which will likely lead to a lower grade on the assignment than you deserve. So please make sure to test your code on the CS servers well ahead of time, so that you can debug and address any problems that might arise.

To help you verify that your code satisfies our specifications, we will prepare a simple test program which we will release soon. The test program will run your program and indicate whether it can understand the output. Note that it will not verify that your solution is correct, just that the output format is correct.

Python versions. Unfortunately, there are two major versions of Python in common use, Python 2.7 and 3.4, These versions are very similar but incompatible enough that most programs written for one version will not work with the other. Despite that Python 3 is now almost 10 years old, Python 2.7 is still about twice as popular,[1] more widely supported, and is the version most students have learned. Our skeleton code is thus designed and tested with Python 2.7 and we recommend using this version in your code.

If you would like to use Python 3.4 instead, please change the first line of your .py file to:

```
#!/usr/bin/env python3
```

and make sure to test on the SOIC Linux machines. If you don't do this, we are likely to get syntax errors when we run your code, the grader will likely assume that your code does not work at all.

Grading. Your code should be correct and clear. Please use comments and meaningful variable names so that we can understand it. In addition, we will use running time as one factor of your grade. For example, for the 10-queens problem, your code should take less than about 30 seconds (and far faster implementations are possible).

Again: it is *very* important that your program satisfy the requirements given above on input format, output format, Python version, etc. It is thus **crucial** that you test your code on the SOIC Linux servers. We will take off points if we cannot run your code, or if we have to modify it in any way to get it to run on the SOIC Linux systems.

What to turn in

Turn in three files via Canvas: (1) a PDF or text document that answers questions 1, 2, 3 and 4, (2) your modified version of nrooks.py, and (3) your source code for a0.py. **Again** (for the third time!), make sure that: (1) your code runs correctly on the CS Linux servers, `burrow.soic.indiana.edu` (aka silo), since this is where we'll test your code; (2) your code has the correct file name and can be run in the required format. *We will deduct points if we have to modify your code in order to make it run on our system.* (Every year, some students ignore this, which both greatly increases grading time and greatly decreases their grade. :-)

[1] `http://www.i-programmer.info/news/98-languages/8269-python-2-versus-python-3-revisited.html`