CpSc 2120: Algorithms and Data Structures

Instructor: Dr. Brian Dean
Webpage: `http://www.cs.clemson.edu/~bcdean/`
Handout 14: Lab #8

Fall 2018
TTh 12:30-1:45
Earle 100

1 Using Recursion to Solve the N-Queens Problem

In this lab exercise, we will hone our recursions skills while solving the famous n-queens problem. Specifically, two queens on a chess board can attack each-other if one piece can reach the other by moving an arbitrary number of squares either horizontally, vertically, or diagonally. The n-queens problem asks us to place n queens on an $n \times n$ chess board so that no queen is attacking any other queen. An example of a solution to the 4-queens problem is below.

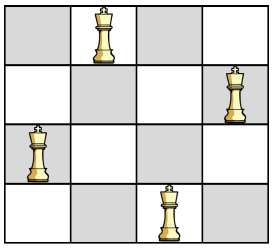

Figure 1: A solution to the 4-queens problem.

There can be many solutions to the n-queens problem. For $n = 8$, for example, there are 92 distinct solutions. Your task with this lab is to compute the number of solutions for any small value of n as quickly as possible.

To count all possible solutions, you should generate them recursively. For example, note that there must be exactly one queen in each row. We can therefore generate solutions row-by-row. That is, we loop through all possible locations for the queen in the first row, then recursively complete the board by checking for each of these all possible locations in the second row, and so on. In code, there are several ways we can write this; for instance, we can maintain the current state of the board in a global (possibly 2D) array, and write a function `check_row(int r)` that tries all possibilities for row r one by one, for each of them calling `check_row(r+1)` to recursively complete the board. You should feel welcome to experiment with different approaches to try and find the fastest one.

In order to achieve as much speed as possible, it will be important to *prune* your recursive search as early as possible. That is, any time you reach a state where two queens are attacking each-other, don't recurse any further, but rather exit the current function and continue searching other alternatives.

Please also exploit the fact that the problem is symmetric to cut down on the search time by a factor of two, as discussed in lecture, by only trying half of the possible locations for the queen in the top row (be careful here if n is odd!)

Your program should take a single integer n as input on the command line and print out a single integer specifying the number of distinct solutions to the n-queens problem.

2 Grading

For this lab, you will receive 5 points for correctness, 2 points for well-written code, and up to 3 points for the speed of your code. Zero points will be awarded for code that does not compile, so make sure your code compiles on the lab machines before submitting!

Final submissions are due by 11:59pm on the evening of Tuesday, October 30. No late submissions will be accepted.