CS-203 – ASSIGNMENT 1 – 2022 SUMMER

GIUSEPPE TURINI – KETTERING UNIVERSITY

INTRODUCTION

N-Queens Problem

The 8-queens problem is a puzzle, originally proposed in 1848 by chess player Max Bezzel. The puzzle asks the player to place 8 queens on a traditional 8×8 chessboard so that no 2 queens are attacking one another. That is, the objective is to select 8 positions in an 8×8 square grid such that no 2 positions appear on the same row, column, or diagonal. [1]

The n-queens problem is a generalization of the 8-queens problem, in which (given a positive integer n) the goal is to place n queens on an n×n square grid such that no 2 queens are attacking one another. The problem has solutions for every n>3. [2]

The original 8-queens problem (n=8) has 92 distinct solutions; the number of solutions drops to 12 if one disregards solutions that are rotations or reflections of the original board. As an example, below (in Figure 1) you can see the only symmetrical solution (excluding rotation and reflection) to the 8-queens' problem.

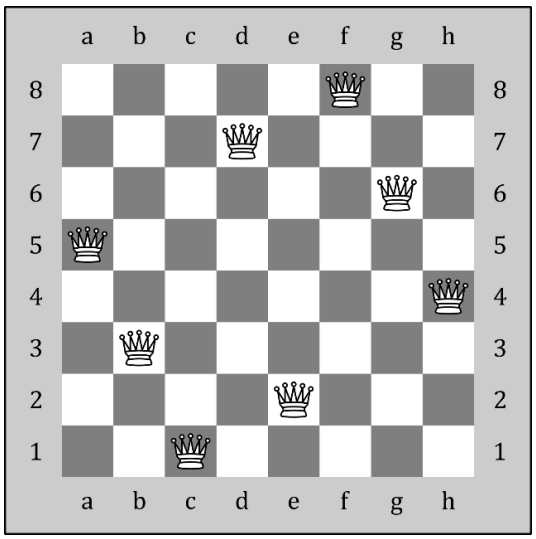

Figure 1. The only symmetrical solution (excluding rotations/reflections) to the 8-queens problem.

[1] See: Wikipedia – Eight Queens Puzzle.

[2] See: Google Optimization Tools – The N-Queens Problem.

Solution by Exhaustive Search

The n-queens problem can be easily solved by an exhaustive search algorithm. Then optimizations can be made to reduce its running time; for example, by observing that any correct solution has exactly 1 queen in each row, and 1 queen in each column.

Solution by Iterative Repair

Another approach to solving the n-queens problem is the *"iterative repair"* algorithm. This program begins by placing 1 queen in each column, using random rows. If this placement yields a solution, the algorithm halts. If not, the algorithm perform column-swaps to reduce the *"attacks"* with the goal of reaching 0 attacks, when a solution is reached. [3]

ASSIGNMENT

Implement an Exhaustive Search Algorithm to Solve the N-Queens Problem

Write a Java program which, given in input n (n>3), finds a solution to the n-queens problem by exhaustive search, and prints the result to standard output.

Implement an Iterative Repair Algorithm to Solve the N-Queens Problem

Write a Java program which, given in input n (n>3), finds a solution to the n-queens problem by iterative repair (see attachment), and prints the result to standard output. [4]

Analyze the Time Efficiency of your Algorithms

Analyze your two algorithms performing both the theoretical analysis and the empirical analysis to evaluate the time efficiency. Each analysis should include: (1) an appropriate description of the analysis process (formulas, data, etc.), (2) a graph properly illustrating the results visually (units, approximation function, etc.), and (3) brief conclusions commenting the results. These analyses should be included in the technical report associated with this assignment, together with a final comparison/discussion of the efficiencies of your algorithms.

[3] See: Wikipedia – Iterative Repair Algorithm.

[4] Sosic R, Gu J. *"A Polynomial Time Algorithm for the N-Queens Problem."* ACM SIGART Bulletin 1, no. 3, pp. 7-11. 1990.

SUBMISSION

Deadline and Procedure

Before Monday of 6th week, send an email to the instructor (gturini@kettering.edu):

- Send the email using your Kettering email account (subject: "CS-203: Assignment 1").
- The email must include in attachment your assignment (2 files, see below).
- Late submissions will be assessed a 3% penalty (-3 points) for each day of delay. [5]

Content

The content of your submission must be:

- A Java file ("SolverNQueens.java") with your code.
- A Technical Report ("CS-203-A1-Report.pdf") with your analyses (~3 pages in total).

EVALUATION

This is the form used to grade this assignment.

ASSIGNMENT 1 – EVALUATION FORM

Implementation, Performance, and Analysis (70/100)
Exhaustive Search Implementation/Performance (20/100)
Exhaustive Search Analyses (10/100)
Iterative Repair Implementation/Performance (30/100)
Iterative Repair Analyses (10/100)
Design, Style, and Submission (30/100)
OOP Design and Compilation Requirements (…/100)
Coding Style and Commenting (…/100)
Submission Procedure and Delay (…/100)

[5] Note: the maximum penalty for a late submission is -30 points.