CS-203 – PROJECT 1 – 2025 SPRING

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.

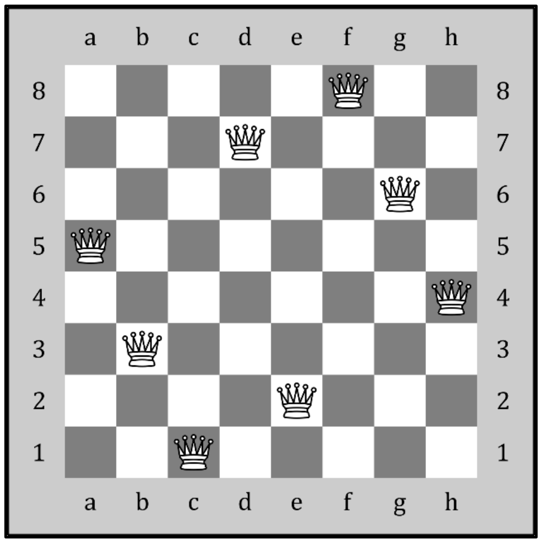

Figure 1. The only symmetrical solution (excluding rotations/reflections) to the 8-queens problem.

[1] See: Wikipedia – Eight Queens Puzzle.

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

Solution by Exhaustive Search or Brute Force

The n-queens problem can be easily solved by an exhaustive search algorithm or a brute force algorithm (these solutions will be discussed and partially implemented in class). Then optimizations can be made to reduce the running time of these simple algorithms (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]

THE PROJECT

Iterative Repair Implementation

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]

Theoretical Analysis of Time Efficiency

Analyze your algorithm implementation performing the theoretical analysis to evaluate its time efficiency, including:

1 Determination of input size and basic operations, with rationale.

2 Description of general best-case, general worst-case, and most-common-case.

3 Definition of the basic operation count for each case (see above), as: summation, closed-form formula, and efficiency class using proper notation and including a brief rationale.

4 One graph illustrating the results of your analysis.

[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.

Empirical Analysis of Time Efficiency

Analyze your algorithm implementation performing the empirical analysis to evaluate its time efficiency, including:

1 Experiment configuration: HW-SW setup, samples size and type, timing vs counting, filtering outliers, etc.

2 One table illustrating the raw data.

3 One graph illustrating the results of your analysis, including the approximation function.

4 A brief explanation of your results, including a summation representing your finding.

Technical Report

Your algorithm implementation, as well as both analyses should be described in a techical report,including:

1 Title matter (student, university, course, instructor, date).

2 A brief description of your implementation, with focus on deviations and bugs.

3 At least one page describing your theoretical analysis.

4 At least one page describing your empirical analysis.

5 A brief comparison of your theoretical and empirical results.

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: Project 1”).
- The email must include in attachment your project (2 files, see below).
- Late submissions will be assessed a -5% penalty (-5 points) for each day of delay. [5]

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

Content

The content of your submission must be:

- A Java file (“SolverNQueens.java”) with your code.
- A technical report (“CS203-Project1-Report.pdf”) with your analysis.

EVALUATION

This is the form used to grade this project.

PROJECT 1 – EVALUATION FORM

Implementation, Performance, and Analysis (70/100)

Iterative Repair Implementation (25/100)
Proper internal data representation, original algorithm structure, no deviations.

Iterative Repair Performance (15/100)
Proper internal data representation, attack data updated in constant time.

Theoretical Analysis (15/100)
Input size, best- and worst- cases, basic operation counts, efficiency classes.

Empirical Analysis (15/100)
Experiment setup, data collection, data visualization, proper result comparison.

Design, Style, and Submission (30/100)

OOP Design and Compilation Requirements (…/100)
Classes, fields, functions, variables properly designed, no unhandled exceptions.

Coding Style and Commenting (…/100)
File/class/function headers, variables comments, comment/code ratio, naming, indentation.

Submission Procedure and Delay (…/100)
Proper submission (email, subject, attachments), no delay.