Green University of Bangladesh

Department of Computer Science and Engineering (CSE)
Semester: (Spring, Year: 2025), B.Sc. in CSE (Day)

N-Queens Visualizer Using Python and Tkinter

Course Title: Artificial Intelligence Lab
Course Code: CSE 316
Section: 221 D7

Student Details

Name	ID
Mohammad Masum Jia	221002085

Submission Date: 10/05/2025
Course Teacher's Name: Md. Sabbir Hosen Mamun

[For teachers use only: Don't write anything inside this box]

Lab Project Status

Marks:	**Signature:**
Comments:	**Date:**

Contents

Chapter 1

Introduction

1.1 Overview

This project presents a Python-based visualization of the classical N-Queens problem. The application allows users to interactively place queens on a dynamically sized chess-board and visualizes the backtracking algorithm used to solve the problem. Built using Tkinter, the visualizer combines educational insights with intuitive GUI features.

1.2 Motivation

Backtracking is a fundamental technique in computer science, but its abstract nature can be challenging for beginners. This project aims to provide a step-by-step, visual understanding of how backtracking algorithms work by solving the N-Queens problem in an engaging way.

1.3 Problem Definition

1.3.1 Problem Statement

The N-Queens problem is to place N queens on an N×N chessboard such that no two queens attack each other. This project implements a real-time visual solver using Python, where each step of the backtracking algorithm is animated to help users understand the logic behind recursive decision making.

1.3.2 Complex Engineering Problem

1.4 Design Goals/Objectives

- Provide an educational visual tool to understand backtracking.

Table 1.1: Summary of the attributes touched by the mentioned projects

Name of the P Attributes	Explain how to address
P1: Depth of knowledge required	Requires knowledge of Python, Tkinter, recursive algorithms, and GUI principles.
P2: Range of conflicting requirements	Balancing between algorithm correctness and visualization complexity.
P3: Depth of analysis required	—-
P4: Familiarity of issues	Understanding of chessboard constraints and queen threats.
P5: Extent of applicable codes	Backtracking algorithm and GUI code must be integrated seamlessly.
P6: Extent of stakeholder involvement and conflicting requirements	—–
P7: Interdependence	Visual, logic, and user interaction components are tightly connected.

- Allow user to interact with board via manual placements.
- Animate each step of recursive placement/removal.
- Make board size dynamic for wider experimentation.

1.5 Application

1.5.1 Education

Used to teach algorithms and data structures—especially backtracking.

1.5.2 Visualization Aid

Helpful in debugging and concept explanation for competitive programmers.

1.5.3 Interactive Experimentation

Allows users to manually place queens and see if their placements lead to a solution.

Chapter 2

Design/Development/Implementation of the Project

2.1 Introduction

This chapter discusses how the N-Queens Visualizer was designed and developed using the Tkinter library and Python logic. Key architectural elements, design principles, and technical implementation are covered.

2.2 Project Details

2.2.1 GUI Layout

The GUI is constructed using ‘tkinter.Canvas‘ for the board and ‘tk.Frame‘ for control buttons. Event listeners are bound to each cell.

2.2.2 Backtracking Logic

The algorithm follows a generator-based recursive backtracking approach that yields steps for animation.

2.2.3 Tools and Libraries

- Python 3
- Tkinter
- Pillow (PIL) for loading queen images

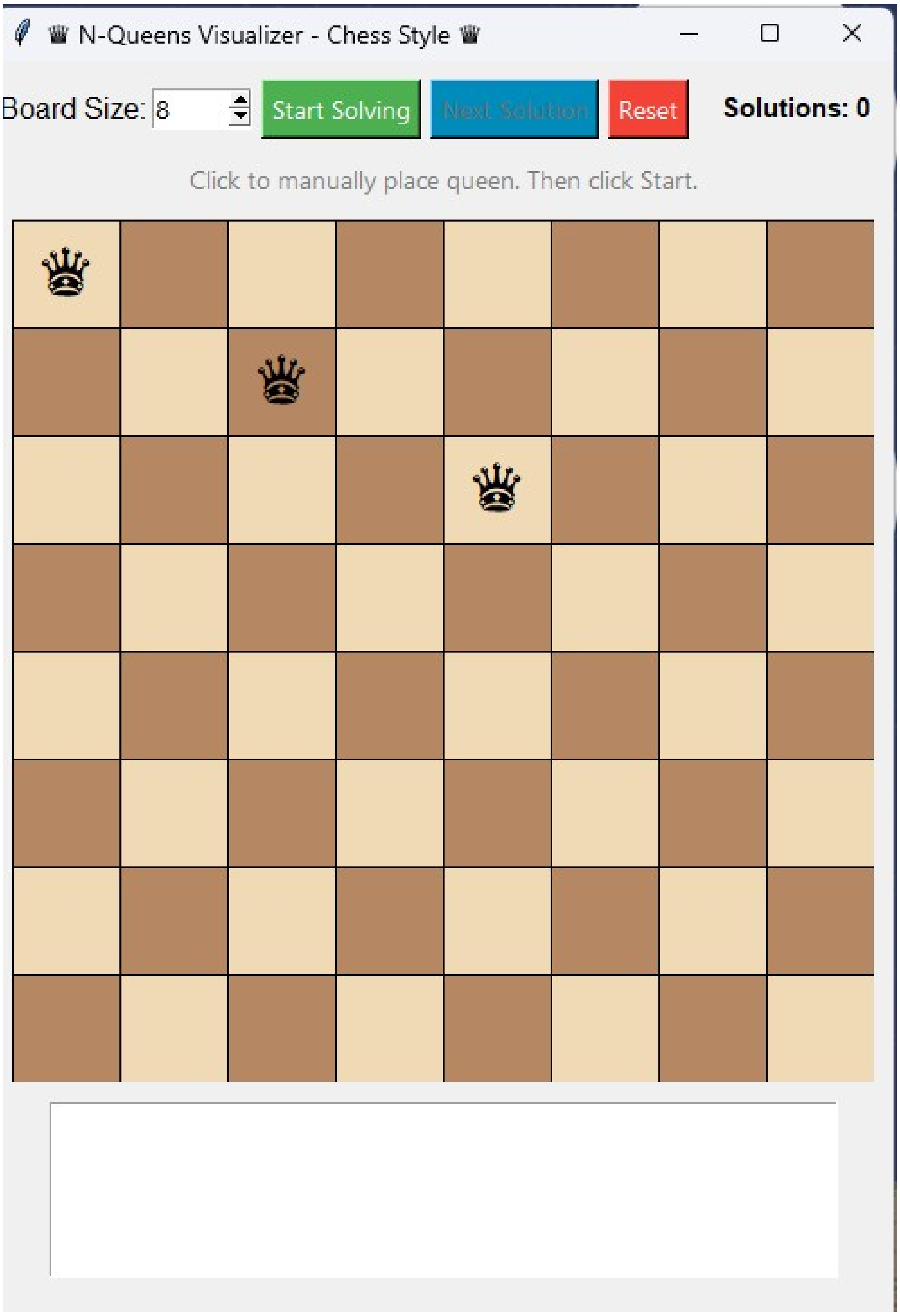
♛ N-Queens Visualizer - Chess Style ♛
Board Size: 8
Start Solving
Next Solution
Reset
Solutions: 0
Click to manually place queen. Then click Start.

Figure 2.1: Board and Control Interface

2.2.4 Implementation Details

GUI and backend logic communicate via event-driven callbacks. Animation is handled using the ‘.after()‘ method in Tkinter to simulate step-by-step solving.

2.3 Algorithms

Algorithm 1: Recursive Backtracking for N-Queens

```
1 Board size N, Initial queen positions (optional) All valid board configurations
  for each row r do
2     for each column c do
3         if queen can be placed at (r, c) then
4             place queen at (r, c)
5             recurse to next row
6             remove queen from (r, c) on backtrack
```

Chapter 3

Performance Evaluation

3.1 Simulation Environment

Tested on Python 3.10 with Tkinter in Windows OS. Required packages include ‘Pillow‘.

3.2 Results Analysis/Testing

3.2.1 Manual Testing

The board was manually tested for different sizes ranging from 4x4 to 12x12. Queens were manually placed in valid and invalid positions to test the safety logic.

3.2.2 Solver Animation

The animated solver was tested to ensure it correctly visualizes the backtracking algorithm. The step delay allowed clear tracking of recursive steps, placements, and removals.

3.2.3 Edge Case Testing

- Board sizes below 4 (e.g., 2x2) correctly return no solution.
- If queens are manually placed in a way that prevents solving, the app correctly shows a "No Solution" message.
- Dynamic resizing resets the board properly.

3.2.4 Edge Case Testing

- Board sizes below 4 (e.g., 2x2) correctly return no solution.

- If queens are manually placed in a way that prevents solving, the app correctly shows a "No Solution" message.
- Dynamic resizing resets the board properly.

3.3 Results Overall Discussion

The project successfully visualizes the N-Queens backtracking algorithm. It allows both automatic and manual testing. All features such as cell clicking, queen placement safety checks, and solution animation worked as intended.

While the application works smoothly up to 12x12 boards, performance starts slowing slightly for larger board sizes due to recursive depth and UI refresh delay.

This tool proves effective for educational purposes and demonstrates core concepts in algorithm design and GUI programming.

Chapter 4

Conclusion

4.1 Discussion

The project achieved its goal of visualizing backtracking through a user-friendly and interactive platform. It bridges the gap between theory and practice, especially in academic contexts.

4.2 Limitations

- Performance drops slightly on large boards ($N > 12$).
- No save/load state functionality yet.
- No support for saving solution images/screenshots.

4.3 Scope of Future Work

- Add timer/speed slider for animation.
- Introduce sound/alert on solution found.
- Provide statistics on steps taken and time used.

References

- Tkinter Docs: https://docs.python.org/3/library/tkinter.html
- Python Official: https://www.python.org/doc/
- Pillow Image Library: https://pillow.readthedocs.io/