আন্তর্জাতিক ইসলামী বিশ্ববিদ্যালয় চট্টগ্রাম
الجامعة الإسلامية العالمية شيتاغونغ
International Islamic University Chittagong

Department of Computer Science and Engineering (CSE)
Faculty of Sciences and Engineering
Semester: (Autumn - 2024), B.Sc. in CSE

"Algorithm Project Report"

Project Title: The N-Queens Interactive Game

Course Title: Algorithm Lab
Course Code: CSE 2422
Section: 4AM

Student Details

	Name	ID
01	Gazi Shaharabi Anwar Tuhin	C231035
02	Minhaz Ahmmed	C231011
03	Shuvo Nath	C231021

Submission Date : 16 Jan 2025
Course Teacher's Name : G.M. Al-Arafat Shaown
Invigilator : Jamil As-Ad

Access Project(GitHub)

[For Teachers use only: Don't Write Anything inside this box]

Project Report Status

Remarks : …… **Signature: .....**

Comments : …… **Date: .....**

N-Queens Interactive Game

Table of Contents

Chapter 1: Introduction

1.1 Introduction

The **N-Queens Interactive Game** project is inspired by the classic **N-Queens problem**, a combinatorial puzzle where the task is to place N queens on an N x N chessboard so that no two queens threaten each other. This means no two queens can be placed in the same row, column, or diagonal.

Our interactive game allows players to explore this puzzle through a dynamic interface where they can test their logic and problem-solving skills. This project aims to create an engaging, educational experience that highlights the intricacies of the N-Queens problem.

1.2 Design Goals/Objectives

The primary objectives of this project are:

- To design an **interactive and educational platform** for solving the N-Queens problem.
- To implement a backtracking-based algorithm that ensures proper validation of queen placements.
- To provide real-time feedback and scoring to enhance the user's experience.
- To visually display incorrect placements and prompt users to improve their strategy.
- To encourage learning through trial and error with a gamified approach.

Chapter 2: Design/Development/Implementation of the Project

2.1 Algorithm

The game logic uses a **backtracking algorithm** to validate the placement of queens. The steps are as follows:

1. Start with an empty chessboard.
2. Place a queen in the first column.
3. For each subsequent column:
    - Check all rows for safe placement.
    - If safe, place the queen and move to the next column.
    - If no safe row is found, backtrack to the previous column and try the next row.
4. Continue until all queens are placed or backtracking determines no solution is possible.

2.2 Snapshots

Game Interface

1. **Game Start Screen**
    - Players can choose between 4-Queens and 8-Queens modes using buttons.
    - Displays current score and a grid-based chessboard.

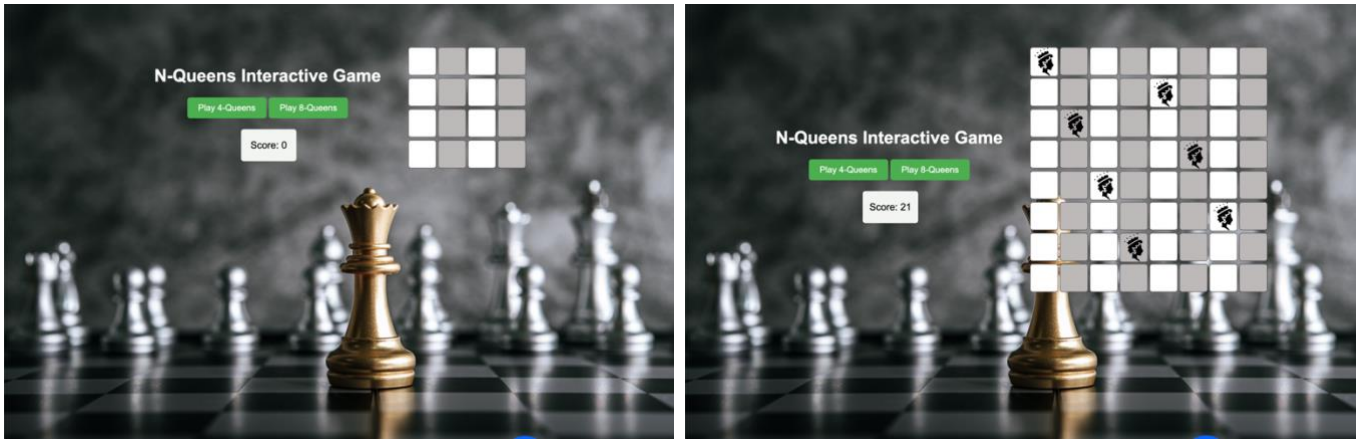

Figure 1: Start Screen with Options for 4-Queens and 8-Queens

2. **Gameplay**
    - Players click cells to place queens, with real-time feedback on incorrect placements.
    - Highlighted cells indicate queens and show conflicts if placement is unsafe.

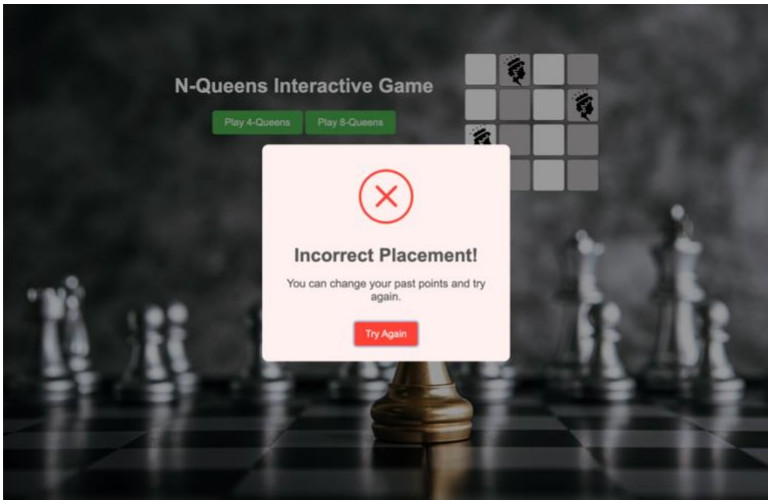

Figure 2: Gameplay Screen with a Queen Placement and Highlighted Conflicts

3. **Victory Screen**
    - After successfully placing all queens, a congratulatory message is displayed.

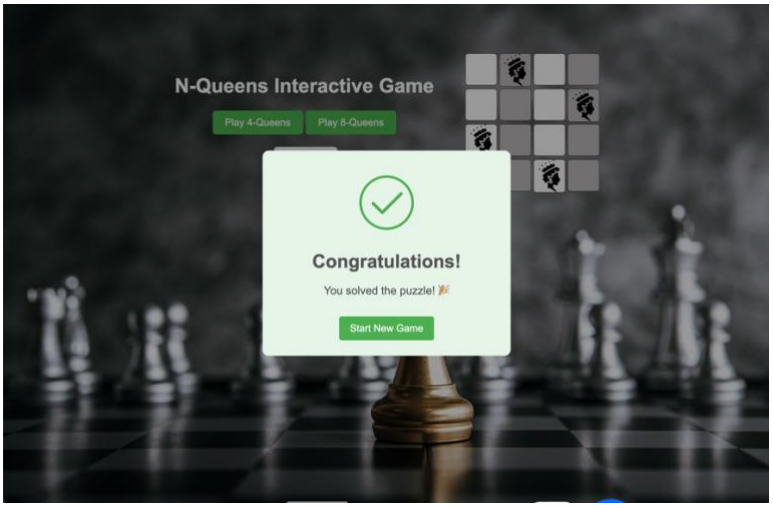

Figure 3: Victory Screen with "Start New Game" Option

2.3 Source Code

"HTML"

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>N-Queens Game</title>
    <link rel="stylesheet" href="style.css">
    <link href="https://cdn.jsdelivr.net/npm/sweetalert2@11.7.3/dist/sweetalert2.min.css" rel="stylesheet">
```

```
</head>
<body>

    <div class="banner">
        <div>
            <h1>N-Queens Interactive Game</h1>
            <div class="button-container">
            <button onclick="startGame(4)">Play 4-Queens</button>
            <button onclick="startGame(8)">Play 8-Queens</button>
            </div>
            <div class="score-board">
            <p>Score: <span id="score">5</span></p>
            </div>
        </div>

        <div>
            <div id="gameBoard" class="game-board" style="--grid-size: 4;"></div>
        </div>
    </div>


    <script src="https://cdn.jsdelivr.net/npm/sweetalert2@11.7.3/dist/sweetalert2.all.min.js"></script>
    <script src="script.js"></script>
</body>
</html>
```

"CSS"

```
body {
    font-family: Arial, sans-serif;
    background: url(images/bg.png) no-repeat center center fixed;
    background-size: cover;
    text-align: center;
    margin: 0;
    padding: 0;
}
h1 {
    color: white;
}
.game-board {
    display: grid;
    margin: 20px auto;
    grid-template-columns: repeat(var(--grid-size), 50px);
    gap: 5px;
```

```
    width: fit-content;
}
.banner
{
    display: flex;
    justify-content: center;
    align-items: center;
    gap: 50px;
    margin-top: 5%;
}
.cell {
    width: 50px;
    height: 50px;
    background-color: white;
    border: 1px solid #555454;
    position: relative;
    cursor: pointer;
    border-radius: 5px;
}
.cell:nth-child(even) {
    background-color: rgb(189, 185, 185);
}
.queen {
    width: 100%;
    height: 100%;
    background-image: url(images/queen.png);
    background-size: cover;
    background-position: center;
}
.score-board {
    padding: 1px;
    font-size: 18px;
    cursor: pointer;
    border: none;
    border-radius: 5px;
    background-color: #f2f5f2;
    color: black;
    max-width: 100px;
    margin-left: 160px;
}
.button-container {
    margin: 20px;
}
button {
    padding: 10px 20px;
```

```
    font-size: 16px;
    cursor: pointer;
    border: none;
    border-radius: 5px;
    background-color: #4caf50;
    color: white;
}
button:hover {
    background-color: #45a049;
}
```

"JavaScript"

```
let score = 0;
let queens = [];

function startGame(size) {
    const board = document.getElementById('gameBoard');
    board.style.setProperty('--grid-size', size);
    board.innerHTML = '';
    score = 0;
    queens = [];
    updateScore();

    for (let row = 0; row < size; row++) {
        for (let col = 0; col < size; col++) {
            const cell = document.createElement('div');
            cell.classList.add('cell');
            cell.dataset.row = row;
            cell.dataset.col = col;
            cell.addEventListener('click', () => placeQueen(row, col, size));
            board.appendChild(cell);
        }
    }
}
function placeQueen(row, col, size) {
    const existingQueenIndex = queens.findIndex(([qRow, qCol]) => qRow === row && qCol === col);
    const cell = document.querySelector(`.cell[data-row='${row}'][data-col='${col}']`);

    if (existingQueenIndex !== -1) {
        queens.splice(existingQueenIndex, 1);
        cell.classList.remove('queen');
        score = Math.max(0, score - 3);  // Prevent negative score
        updateScore();
        return;
```

```
}

if (!isSafe(row, col)) {
    Swal.fire({
        title: 'Incorrect Placement!',
        text: 'You can change your past points and try again.',
        icon: 'error',  // Red error icon
        confirmButtonText: 'Try Again',
        confirmButtonColor: '#FF4136',  // Red button color
        background: '#fff0f0',  // Light red background
        iconColor: '#FF4136',  // Red icon
        customClass: {
            popup: 'smaller-rectangular-alert',  // Custom class for rectangular popup
            title: 'font-semibold text-xl',  // Adjusted text size for title
            text: 'text-base',  // Adjusted text size for message
            confirmButton: 'px-6 py-2 rounded-lg text-white'  // Adjust button padding
        },
        willOpen: () => {
            const popup = Swal.getPopup();
            popup.style.width = '400px';  // Custom width for rectangular shape
            popup.style.borderRadius = '10px';  // Make it rectangular with rounded corners
        }
    });

    score = Math.max(0, score - 3);  // Prevent score going below zero
    updateScore();
    return;
}

cell.classList.add('queen');
queens.push([row, col]);

score += 3;  // Increase score on correct placement
updateScore();

if (queens.length === size) {
    Swal.fire({
        title: 'Congratulations!',
        text: 'You solved the puzzle! 🎉',
        icon: 'success',  // Green success icon
        confirmButtonText: 'Start New Game',
        confirmButtonColor: '#4CAF50',  // Green button color
        background: '#e8f5e9',  // Light green background
        iconColor: '#4CAF50',  // Green icon
```

```
            customClass: {
                popup: 'smaller-rectangular-alert',  // Custom class for rectangular popup
                title: 'font-semibold text-xl',  // Adjusted text size for title
                text: 'text-base',  // Adjusted text size for message
                confirmButton: 'px-6 py-2 rounded-lg text-white'  // Adjust button padding
            },

            willOpen: () => {
                const popup = Swal.getPopup();
                popup.style.width = '400px';  // Custom width for rectangular shape
                popup.style.borderRadius = '10px'; //Make it rectangular with rounded corners
            }
        }).then(() => {
            startGame(size);  // Start a new game after the user confirms
        });
    }
}
function updateScore() {
    document.getElementById('score').textContent = score;  // Ensure score display is updated properly
}
function isSafe(row, col) {
    for (const [qRow, qCol] of queens) {
        if (qRow === row || qCol === col || Math.abs(qRow - row) === Math.abs(qCol - col)) {
            return false;
        }
    }
    return true;
}
function updateScore()
 {
    document.getElementById('score').textContent = score;
    if (score <= -1) {
        alert('Game Over! You ran out of points.');
        startGame(queens.length);
    }
}
startGame(4);
```

Chapter 3: Performance Evaluation

3.1 Results and Discussions

1. **Interactive Gameplay**:

    - The game interface successfully allows users to place queens and receive immediate feedback.
    - Incorrect placements are highlighted, improving user engagement and understanding.
2. **Algorithm Performance**:
    - The backtracking algorithm effectively detects unsafe placements, ensuring only valid solutions are accepted.
    - Players can learn the N-Queens problem intuitively through repeated attempts.
3. **Scoring System**:
    - The scoring mechanism incentivizes strategic thinking by penalizing incorrect placements.
    - Real-time updates maintain user focus and motivation.

Chapter 4: Conclusion

4.1 Introduction

This project demonstrates the applicability of the backtracking algorithm in solving a combinatorial problem. The interactive design transforms a theoretical concept into an engaging, hands-on learning tool.

4.2 Practical Implementation

- The game is hosted on a browser-based platform, making it accessible and user-friendly.
- It can be extended to support higher dimensions (e.g., 12-Queens) and additional learning tools.

4.3 Scope of Future Work

- **Algorithm Optimization**: Implement advanced techniques like heuristic search to improve performance.
- **Enhanced UI/UX**: Add animations and more sophisticated visuals.
- **Educational Features**: Include explanations of solutions and challenges for users to attempt larger boards.
- **Mobile Support**: Ensure compatibility with mobile devices for broader accessibility.

Project: https://shaharabituhin.github.io/N-Queen-Algorithm-Iterative-Game/
Reference :
https://en.wikipedia.org/wiki/Backtracking
https://atechdaily.com/posts/Flowchart-and-Algorithm-for-N-Queen-Problem