TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

A PROJECT REPORT ON

APPLICATION OF DATA STRUCTURE AND ALGORITHM IN SUDOKU SOLVER

By:

Nishan Adhikari (079BCT050)

Prabesh Dhakal (079BCT057)

Sudhan Bhattarai (079BCT082)

Vaibhav Jhunjhunuwala (079BCT095)

A PROJECT REPORT TO THE DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING ON DATA STRUCTURE AND ALGORITHM

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

LALITPUR NEPAL

FEBRUARY 2025

ACKNOWLEDGEMENT

We express our heartfelt gratitude to our Data Structures and Algorithms lecturer, Bibha Sthapit, for her invaluable guidance and support throughout this project. Her insights into algorithmic problem-solving have been instrumental in shaping our understanding and implementation of recursion and backtracking.

We extend our appreciation to the Department of Electronics and Computer Engineering at Pulchowk Campus for providing us with the opportunity and resources to undertake this project. Their encouragement in applying theoretical knowledge to practical applications has been invaluable.

Our sincere thanks go to the developers and contributors of React and TypeScript, whose frameworks and extensive documentation facilitated the development of our Sudoku Solver web application.

We are also grateful to our peers and classmates for their constructive feedback and discussions, which helped refine our approach and enhance our learning experience.

Lastly, we express our deepest appreciation to our families for their unwavering support and patience during this project. Their encouragement motivated us to push forward and complete this work successfully.

We welcome any feedback or suggestions for improvement, as they will help us further enhance our understanding of algorithmic problem-solving and software development.

Authors:

Nishan Adhikari

Prabesh Dhakal

Sudhan Bhattarai

Vaibhav Jhunjhunuwala

ABSTRACT

The This project presents the development of a Sudoku Solver web application using React, TypeScript, and Tailwind CSS, with a primary focus on demonstrating key Data Structures and Algorithms (DSA) concepts. The core algorithm utilized is recursive backtracking, which efficiently solves Sudoku puzzles by exploring possible number placements while adhering to constraint satisfaction rules.

The application features an interactive Sudoku grid, allowing users to manually input puzzles or generate new ones with varying difficulty levels. A real-time solution visualization provides a step-by-step demonstration of the backtracking process, enhancing user understanding of how recursion systematically explores and eliminates possibilities. TypeScript ensures code reliability and maintainability, while the modular React-based architecture supports scalability and future enhancements.

This project bridges theoretical DSA concepts with practical implementation, emphasizing algorithmic problem-solving, recursion-based techniques, and UI/UX integration. By providing an interactive and educational tool, the Sudoku Solver serves as a valuable resource for understanding backtracking algorithms and constraint satisfaction problems in real-world applications.

Keywords: Sudoku Solver, React, TypeScript, Backtracking, Recursion, DSA, Visualization.

TABLE OF CONTENTS

1. OBJECTIVES 1

2. INTRODUCTION 2

2.1 Theoretical Background on Object Oriented Programming, C++ and Qt Library 2

3. APPLICATION 3

4. LITERATURE SURVEY 4

4.1. C++ 4

4.1.1. STRUCTURE OF PROGRAM 4

4.2. QT 6 .....**Error! Bookmark not defined.**

4.2.1. Key Features of Qt 6 .....**Error! Bookmark not defined.**

5. EXISTING SYSTEM 6

6. METHODOLOGY 7

7. IMPLEMENTATION 9

7.1. SYSTEM BLOCK DIAGRAM 12

8. RESULT 12

9. PROBLEM FACED AND SOLUTIONS 17

10. LIMITATIONS AND FUTURE ENHANCEMENT 18

11. CONCLUSION AND RECOMMENDATIONS 19

11.1. RECOMMENDATIONS 19

12. REFERENCES 20

1. OBJECTIVES

The primary objective of this Sudoku Solver web application project is to develop a digital tool that efficiently solves Sudoku puzzles using recursion and backtracking algorithms. The system aims to provide an interactive, educational, and user-friendly platform for visualizing and understanding algorithmic problem-solving techniques.

Key objectives include:

- Demonstrate the application of backtracking and recursion in solving constraint satisfaction problems.
- Provide an intuitive interactive interface for users to input Sudoku puzzles and view step-by-step solutions.
- Visualize the solving process to enhance the learning experience of algorithmic techniques.
- Use React and TypeScript to create a reliable, maintainable, and scalable web application.

2. INTRODUCTION

Sudoku is a widely recognized puzzle that provides an excellent opportunity to demonstrate the application of Data Structures and Algorithms (DSA). This project focuses on developing a Sudoku Solver web application that uses recursion and backtracking algorithms to efficiently solve puzzles. The application allows users to input Sudoku puzzles, view step-by-step solutions, and understand the process behind solving such constraint satisfaction problems.

By utilizing React and TypeScript, this project combines modern web development technologies with algorithmic problem-solving. It aims to create an interactive and user-friendly platform where users can visually track the algorithm's progress while learning about key DSA concepts, particularly recursion and backtracking. Through this application, users gain insights into how algorithms work in real-time, enhancing their understanding of problem-solving techniques.

2.1 Theoretical Background on Data Structures and Algorithms, React, and TypeScript

Data Structures and Algorithms (DSA) form the foundation of computer science, focusing on the efficient organization and manipulation of data to solve problems. Data structures like arrays, stacks, queues, trees, and graphs are used to store and organize data, while algorithms provide step-by-step procedures to manipulate this data to solve specific tasks. One of the key algorithms used in this project is backtracking, a form of recursion that is particularly effective in solving constraint satisfaction problems like Sudoku. Backtracking explores potential solutions by systematically testing all possibilities and backtracking when a solution path is found to be invalid. This algorithm is essential in finding the solution to Sudoku puzzles, where the goal is to fill a grid following a set of rules.

React is a widely used JavaScript library for building user interfaces, especially for single-page applications where dynamic content is frequently updated. It allows developers to build component-based UIs, where each component encapsulates its own state and logic, ensuring modularity and reusability. React also optimizes performance with its virtual DOM, ensuring minimal updates to the actual DOM,

which enhances user experience. TypeScript, a superset of JavaScript, adds static typing to the language, which helps in detecting errors early during development and makes the code more maintainable and scalable. TypeScript improves code quality by ensuring that variables and function calls conform to defined types, reducing runtime errors. Together, React and TypeScript provide a powerful framework for building dynamic, high-performance web applications, such as the Sudoku Solver, where users can interact with the app to input puzzles and visualize the solution process.

3. APPLICATION

The Sudoku Solver web application serves as an educational tool for understanding recursion and backtracking algorithms. It helps users visualize algorithmic problem-solving in real-time while solving Sudoku puzzles. Additionally, it demonstrates the practical application of Data Structures and Algorithms (DSA) in web development, offering insights into how such algorithms can be implemented in a user-friendly environment. This project is valuable for learners seeking to understand key DSA concepts and how they are used in solving real-world problems.

4. LITERATURE SURVEY

4.1. Data Structures and Algorithms (DSA)

Data Structures and Algorithms (DSA) form the cornerstone of computer science, focusing on organizing and manipulating data efficiently to solve problems. The foundation of any algorithm-based system lies in choosing the appropriate data structure, which impacts performance and memory usage. The key concepts in DSA that are utilized in this project include recursion and backtracking.

- **Recursion** is a method where a function calls itself to solve smaller instances of a problem, often breaking down complex problems into simpler subproblems.
- **Backtracking** is a specialized algorithmic technique for solving constraint satisfaction problems like Sudoku. It explores all potential solutions incrementally, abandoning a solution as soon as it is determined to be invalid (i.e., "backtracking"). This project uses these techniques to solve Sudoku puzzles by filling the grid step-by-step, while ensuring all constraints are satisfied.

4.1.1. Recursive Backtracking Algorithm

In recursive backtracking, the algorithm builds a solution piece by piece and removes those pieces that lead to dead ends. For Sudoku, backtracking attempts to place numbers in empty cells while checking if the current solution violates any constraints. If a conflict arises, the algorithm undoes the last step and tries a new possibility, making it particularly effective for solving puzzles like Sudoku. This method is both intuitive and efficient for constraint satisfaction problems, as it efficiently prunes invalid solutions, reducing the search space.

4.2. React and TypeScript

React enables efficient, component-based UI development with a virtual DOM for performance optimization and a declarative syntax for easier UI management. TypeScript enhances React by adding static typing, improving error detection during development, and offering better tooling support with features like auto-completion and refactoring, making code more maintainable and reliable. Together, they provide a scalable, high-performance framework for building interactive and robust web applications.

4.2.1. Key Features of React and TypeScript

- **Component-Based Architecture (React):** Allows modular and reusable UI components, improving code maintainability.
- **Virtual DOM (React):** Optimizes UI rendering by updating only necessary elements, enhancing performance.
- **Static Typing (TypeScript):** Ensures early error detection by enforcing type safety during development.
- **Enhanced Tooling and IDE Support (TypeScript):** Provides features like auto-completion, refactoring, and better code navigation for improved developer productivity.

5. EXISTING SYSTEM

Current systems for solving Sudoku puzzles, both online and offline, often rely on traditional brute-force methods or static solvers that do not visually demonstrate the algorithmic process behind the solution. Many of these systems are either static in nature, where the solution is shown without interaction, or they are basic backtracking solvers with limited user input functionality.

In educational platforms, the main issue is the absence of interactivity and visualization, limiting users' understanding of algorithms like backtracking. While some applications offer features like puzzle generation and solution display, they often overlook the educational value of demonstrating recursion and backtracking.

Our Sudoku Solver web application addresses these gaps by offering an interactive, step-by-step visualization of the solving process, showcasing backtracking techniques, and allowing users to input custom puzzles and adjust difficulty levels for a more engaging learning experience.

6. METHODOLOGY

The methodology for developing the Sudoku Solver web application follows a structured approach to ensure an efficient and effective solution. The process is divided into the following key phases:

1. **Requirement Analysis:**
    - Identify and document the functional and non-functional requirements of the system.
    - Analyze existing Sudoku solvers to identify limitations and user needs.
    - Define the scope, objectives, and key features, including interactive puzzle solving, difficulty levels, and visual algorithm demonstration.
2. **System Design:**
    - Develop the architecture and design of the application, focusing on both front-end and back-end components.
    - Define the data structures for representing Sudoku puzzles and solutions.
    - Create wireframes and prototypes for the user interface (UI) for both puzzle input and solution visualization.
3. **Development:**
    - Set up the development environment using React for the front-end and TypeScript for enhanced code quality and functionality.
    - Implement the back-end logic, including the backtracking algorithm and recursive solving mechanism.
    - Develop the front-end components, focusing on interactive puzzle input, step-by-step solution visualization, and user-friendly controls for adjusting difficulty levels.
    - Integrate the front-end and back-end components to ensure smooth interaction.
4. **Testing:**
    - Develop test cases to cover all functional requirements, ensuring the accuracy of the Sudoku solving process and the proper functioning of interactive features.

- Perform system testing to validate the overall performance and usability of the application.
- Implement security and performance testing to ensure the application runs efficiently and securely for all users.

5. **Deployment:**
   - Prepare the deployment environment, ensuring compatibility across different web browsers and devices.
   - Monitor the system after deployment to address any user feedback and issues that arise.
6. **Maintenance and Support:**
   - Regularly update the system to improve functionality, add new features (e.g., puzzle difficulty enhancements), and resolve any reported bugs.

7. **Tools and Technologies:**
   - Utilize React, TypeScript, and JavaScript for front-end development.
   - Implement the backtracking algorithm in TypeScript for solving the Sudoku puzzle.
   - Use version control with Git for code management and collaboration during development.