Data Structures and Algorithms with Python Internship

An Internship report submitted to

Jawaharlal Nehru Technological University Anantapur, Anantapuramu

In partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING

Submitted by

Y.Yeswanth Reddy

21121A05T6

III B.Tech I Semester

Under the esteemed supervision of

Dr .G . Sunitha

Professor

Department of Computer Science and Engineering

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SREE VIDYANIKETHAN ENGINEERING COLLEGE

(AUTONOMOUS)

(Affiliated to JNTUA, Anantapuramu and approved by AICTE, New Delhi) Accredited by NAAC with A Grade

Sree Sainath Nagar, Tirupati, Chittoor Dist. -517 102, A.P, INDIA

2023 - 2024.

SREE VIDYANIKETHAN ENGINEERING COLLEGE
(AUTONOMOUS)
Sree Sainath Nagar, A. Rangampet

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Certificate

This is to certify that the internship report entitled ***"Data Structures and Algorithms Internship"*** *is the bonafide work done by* ***Y.Yeswanth Reddy****(Roll No:21121A05T6) in the Department of* ***Computer Science and Engineering****, and submitted to Jawaharlal Nehru Technological University Anantapur, Anantapuramu in partial fulfillment of the requirements for the award of the degree of* ***Bachelor of Technology*** *in* ***Computer Science*** *during the academic year 2023-2024 .*

Head:

Dr.G.Sunitha
Professor
Dept,of CSE

Dr. B. Narendra Kumar Rao
Professor & Head
Dept. of CSE

`

INTERNAL EXAMINER

EXTERNAL EXAMINER

COMPLETION CERTIFICATE FROM COMPANY

ABSTRACT

This internship focuses on providing participants with a comprehensive understanding of data structures using the Python programming language. Data structures are fundamental components in computer science, influencing the efficiency and organization of algorithms. Python, with its simplicity and versatility, serves as an ideal platform for exploring these essential concepts.

Interns will delve into various data structures, including but not limited to arrays, linked lists, stacks, queues, trees, and graphs. The program aims to empower participants with hands-on experience in implementing, manipulating, and optimizing these structures in Python, fostering a deep understanding of their underlying principles.

Key Objectives:

1. **Fundamental Concepts:** Participants will gain a solid foundation in the theoretical aspects of data structures, understanding their importance in algorithmic design and problem-solving.
2. **Python Implementation:** The internship will extensively cover the implementation of data structures using Python, leveraging the language's built-in capabilities and libraries.
3. **Hands-on Projects:** Interns will engage in practical projects to reinforce their learning. These projects will involve real-world scenarios, encouraging the application of data structures to solve problems efficiently.
4. **Optimization Techniques:** The program will explore strategies for optimizing code through the judicious selection and use of data structures. Participants will learn to evaluate trade-offs and make informed decisions to enhance program efficiency.
5. **Problem-Solving Skills:** Through coding exercises and challenges, interns will sharpen their problem-solving skills, applying data structures to tackle a variety of computational problems.
6. **Collaborative Learning:** The internship will foster a collaborative environment, encouraging interns to work on team projects, share insights, and learn from each other's experiences.

By the end of the internship, participants will have a strong grasp of data structures, their implementation in Python, and the ability to apply this knowledge to solve complex programming problems. This experience will not only enhance their proficiency in Python but also prepare them for future challenges in computer science and software development.

ACKNOWLEDGEMENT

We are extremely thankful to our beloved Chairman and founder **Dr. M. Mohan Babu** who took keen interest to provide us the oppurtunity for carrying out the project work.

We are highly indebted to **Dr. B.M.Satish**, Principal of Sree Vidyanikethan Engineering College for his valuable support in all academic matters.

We are very much obliged to **Dr. B. Narendra Kumar Rao,** Professor & Head, Department of CSE, for providing us the guidance and encouragement in completion of this work.

I would like to express my special thanks of gratitude to the YBI Foundation, Delhi who gave me the golden opportunity to do this wonderful internship, which also helped me in doing a lot of Research and I came to know about so many new things I am really thankful to them.

Y.Yeswanth Reddy
21121A05T6

TABLE OF CONTENTS

Title		Page no
Completion certificate from company		**i**
Abstract		**ii**
Acknowledgement		**iii**
Table of contents		**iv**
List of Figures		**v**
List of Tables		**vi**
Abbreviations		**vii**
Chapter 1 Introduction		1
1.1	Description of the Company	2
1.2	Overview of the Project	3-20
1.3	Technology Stack	21-24
Chapter 2 Summary of Experience		**25-26**
Chapter 3 Reflection on Learning		**27**
Conclusion		**28-29**

List of Figures

S.No.	Title of figure	Page no
1	Sudoku Format	4
2	Naked Single Method	6
3	Hidden Single Technique	7
4	Naked Pair Technique	8
5	Pencil-and-paper algorithm vs brute force algorithm	11
6	Pencil-and-paper algorithm vs backtracking algorithm	17

List of Tables

S.No.	Title of table	Page no
1	Pencil-and-paper Algorithm vs Brute Force Algorithm	10
2	Methods involve in solving sudoku	15
3	Pencil and Pen algorithm vs Backtracking algorithm	16
4	Backtracking Algorithm vs Pencil-and-paper Algorithm vs Brute-force	19

Abbreviations

- **YBI:** Yanthra Byte Internships Foundation
- **Edutech**: Educational Technology
- **IDE:** Integrated Development Environment
- **CI/CD:** Continuous Integration/Continuous Deployment
- **SQL:** Structured Query Language
- **HTML:** Hypertext Markup Language
- **CSS:** Cascading Style Sheets
- **API:** Application Programming Interface
- **Git:** Version Control System
- **GUI**: Graphical User Interface
- **PDF:** Portable Document Format
- **PNG:** Portable Network Graphics
- **JPEG:** Joint Photographic Experts Group
- **SVG:** Scalable Vector Graphics
- **RAM:** Random Access Memory
- **P&P algorithm:** Paper and Pen algorithm

CHAPTER 1

INTRODUCTION

Introduction to 2-Month Internship on DSA with Python at YBI Foundation

The YBI Foundation's immersive 2-month internship program focused on Data Structures and Algorithms with Python. This internship is designed to provide participants with a comprehensive understanding of fundamental concepts in DSA and enhance their proficiency in the Python programming language.

Program Overview:

- **Duration:** 2 Months
- **Focus Area:** Data Structures and Algorithms
- **Programming Language:** Python

Key Highlights:

Hands-on Learning: Engage in practical, hands-on coding exercises and projects that reinforce theoretical concepts in DSA.

Python Proficiency: Develop a strong command of Python, a versatile and widely-used language known for its simplicity and readability.

Real-world Applications: Apply DSA concepts to real-world problem-solving scenarios, emphasizing practical implementation.

Project Work: Undertake a capstone project, such as building a Sudoku Solver using Python and the backtracking algorithm.

Collaborative Learning Environment:

Experience a collaborative and supportive learning environment, where you'll have the opportunity to work on team projects, share insights with peers, and collectively tackle coding challenges.

1.1Description of the company:

YBI foundation is an online platform for providing virtual internship, skills, training and courses in emerging technologies to students with an opportunity to get completion certificate. We help students to Learn Practice Upskill and Get Job Ready with scientifically designed content by industry experts.

YBI Foundation is a Delhi-based not-for-profit edutech company that aims to enable the youth to grow in the world of emerging technologies. They offer a mix of online and offline approaches to bring new skills, education, technologies for students, academicians and practitioners. They believe in the learning anywhere and anytime approach to reach out to learners. The platform provides free online instructor-led classes for students to excel in data science, data structures, business analytics, machine learning, cloud computing and big data. They aim to focus on innovation, creativity, technology approach and keep themselves in sync with the present industry requirements. They endeavor to support learners to achieve the highest possible goals in their academics and professions.

YBI Foundation stands as a pioneering force in the educational landscape, bridging the gap between traditional learning and the demands of the contemporary job market. With a commitment to empowering youth in the realm of emerging technologies, the edutech company employs a holistic approach that encompasses both online and offline methodologies.

1.2 Overview of the Project:

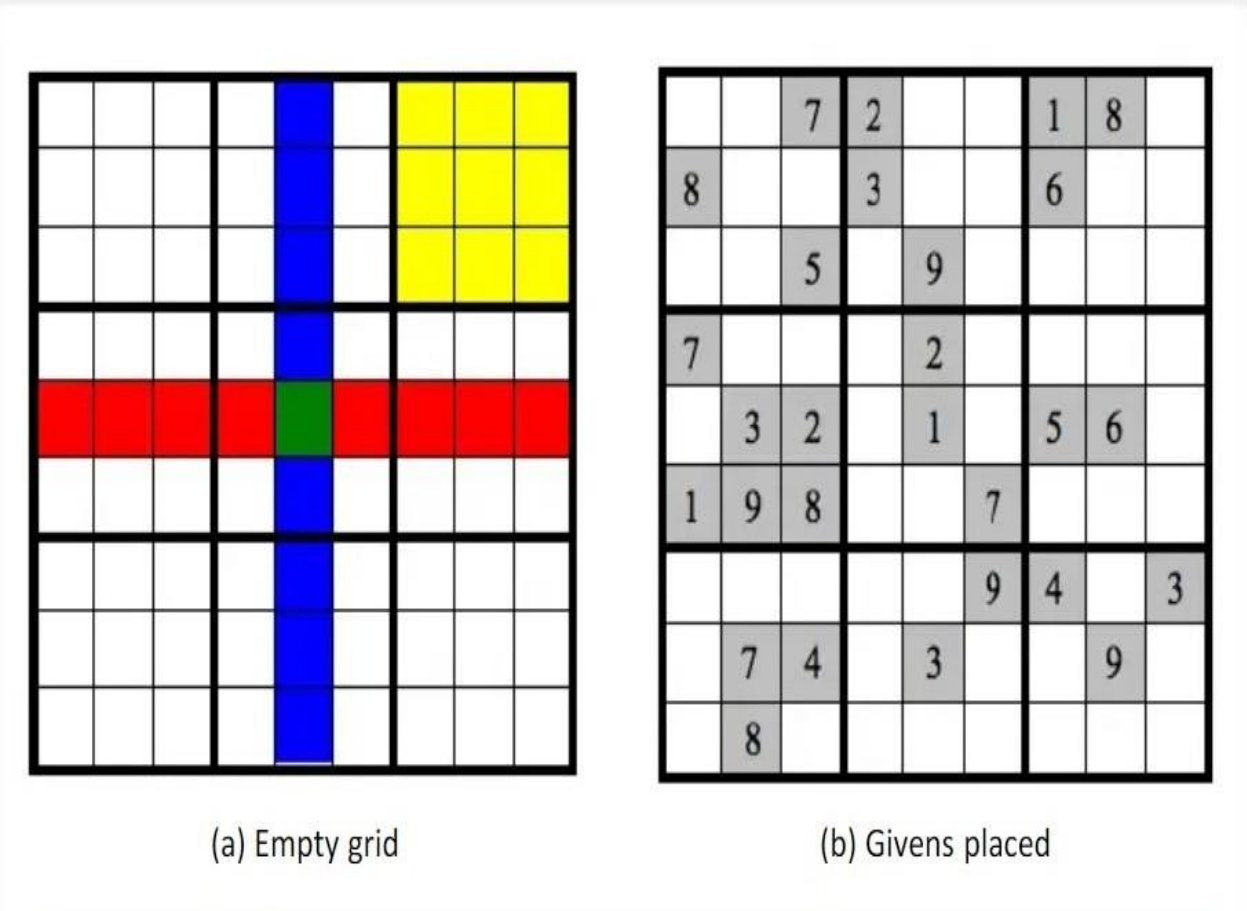
Title: Sudoku Solving Project using Backtracking Algorithm in Python

I. Introduction to Sudoku

Sudoku, a globally popular number puzzle, captivates enthusiasts with its perfect blend of logic and pattern recognition. Originating from Switzerland in the 18th century and later popularized in Japan, Sudoku has become a cultural phenomenon. The game's allure lies in its simplicity – a 9x9 grid divided into 3x3 subgrids that must be filled with digits from 1 to 9, following strict rules to ensure a unique and solvable solution.

Sudoku is a logic-based puzzle that is played by numbers from 1 to 9. The Puzzle first appeared in newspapers in November 1892 in France and then Howard Garns an American architect presented it in its modern form . There are already many journals, papers and essays that researched about Sudoku Solvers and most of them present different type of algorithms. Sudoku's popularity is based on several reasons. First of all it is fun and fascinating, and very easy to learn because of its simple rules. There are currently many different type of Sudoku puzzles, classic Sudoku that contains a 9X9 grid with given clues in various places, mini Sudoku that consists of a grid with 4X4 or 6X6 sizes. The other type of Sudoku is Mega Sudoku that contains a grid with 12X12 or 16X16 sizes . In this text, the focus is mostly on the classic Sudoku, i.e. 9X9 grid. Furthermore, Sudoku has become so popular, compared to other games, all over the world because its rules are easy to understand and it can improve our brain and also it is fun. The structure of the puzzle is very simple, especially the classic puzzle. This essay is mainly focused on classic puzzle of a 9X9 grid. There already exist a number of digits in the board that make the puzzle solvable. It means that some numbers are already placed in the Sudoku board before starting playing. The board consists of 81 cells, which is divided into nine 3X3 sub boards and each 3X3 sub board is called "box" or "region". The main concept of the game is to place numbers from 1 to 9 on a 9X9 board so that every row, column and box contains any numbers but once. This means that no number is repeated more than once.

Generally, the puzzle has a unique solution. There are certain techniques to solve the puzzle by hand and these rules can be implemented into a computer program



Sudoku Format

a.Empty grid: An empty Sudoku grid is typically a 9x9 grid with 9 3x3 subgrids . Each row, column, and 3x3 subgrid should eventually be filled with the numbers 1 through 9, with no repetition in any row, column, or subgrid.

You can use this grid as a starting point to fill in the numbers based on the rules of Sudoku. Remember that each row, column, and 3x3 subgrid must contain the numbers 1 through 9 with no repetition.

b.Givens placed: In a Sudoku puzzle, givens are the initially provided numbers that serve as clues to help you solve the puzzle. They are the numbers pre-filled in the grid, and the challenge is to complete the rest of the grid based on these initial values.

The difficulty level of Sudoku puzzles depends on how the given numbers are placed in the Sudoku board and also how many numbers (clues) are given. Generally, the most significant aspect of difficulty ratings of Sudoku puzzles is that which techniques are required to solve the puzzles. In other words, it is important where the given numbers are placed logically. The Puzzles, which needs more techniques to solve, can be named as difficult one. On the other side there are puzzles that can be solved by using simple methods and this kind of puzzles can be defined as easy or medium level. As mentioned above, there are four difficulty levels that are used in testing (easy, medium, hard and evil). We have found during testing that the classifications of difficulty levels is not easy as it is stated. This is due to the fact that there have been puzzles which marked as hard level but they had been solved using simple techniques and vice versa

A. History

The history of Sudoku traces back to the works of Swiss mathematician Leonhard Euler in the 18th century, but it gained widespread recognition in the 20th century through Japanese puzzle publishers. Over the years, Sudoku has evolved into a global pastime, fostering communities of enthusiasts and even international competitions.

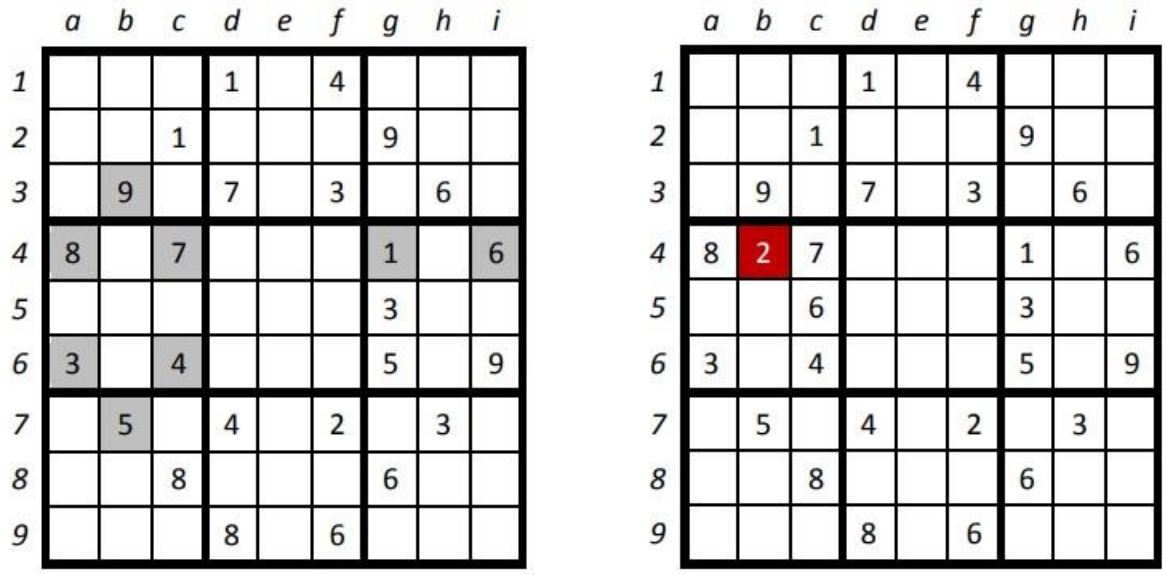
B. Pencil-and-paper algorithm

In this work, we implement a solution based on some strategies used by humans when solving the puzzle, therefore, it is called pencil-and-paper algorithm. The paper-and-pencil algorithm contains human strategies. These strategies have been examined below in more details. These techniques are almost easy to understand by human players, but it might be hard to search in the puzzle, since there are several things to look for at the same time. As there are puzzles with different types of difficulty, the easy and medium puzzles can be solved using some simple

techniques such as unique missing method, naked singles. However to solve difficult problems we may examine other techniques as well (locked candidates, naked and hidden pairs, triplets etc.).

a.Unique missing candidate: The unique missing candidate is used when any row, column or box is missing only one single digit.

b.Naked Singles: This method is useful when we find a square that can only take one single value, once the contents of other squares in the same row, column and box are considered. Additionally, this is when the row, column and box hold 8 different numbers and one single number is left for that square.

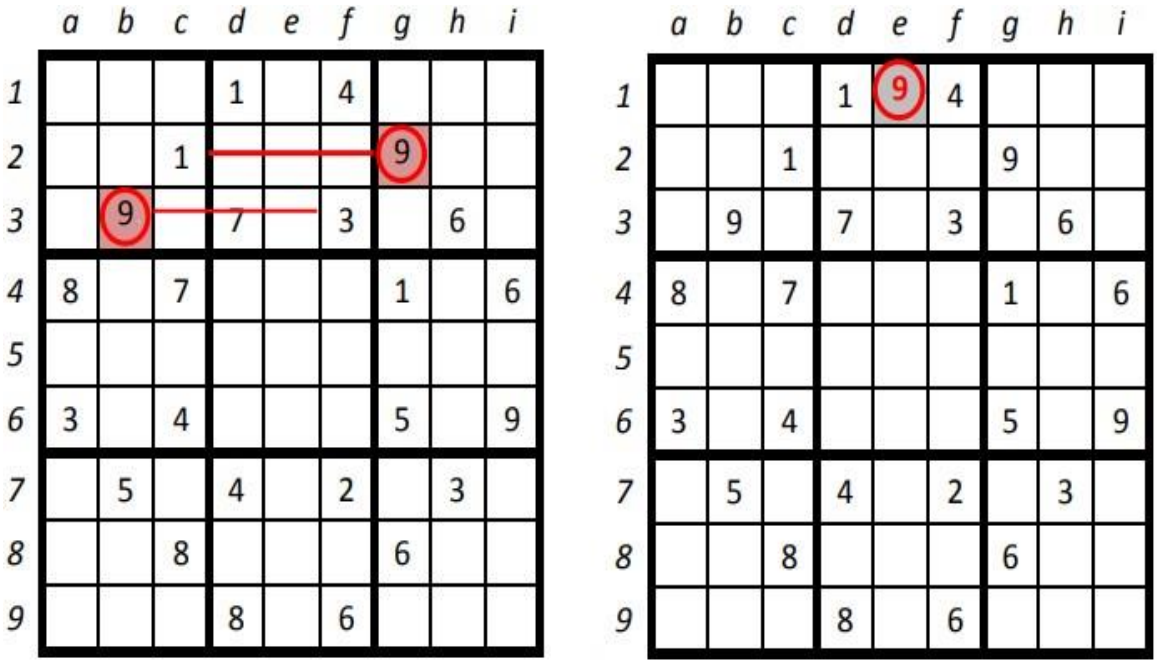


Naked Single Method

A description of the naked single method. In the left figure square 4b can hold just one possible number, which is 2 as it is inserted in the right figure.

As we see in figure , it is possible to list all the candidates from 1 to 9 in each unfilled square, i.e. square 4b can only hold number 2 since it is the only candidate for this position. The most significant aspect is that when a candidate is found for a certain position then it can be removed from the list as a possible candidate in the row, column and box . The reason that it is called the “naked single” method is that this kind of square contains only one possible candidate.

c.Hidden Singles: The hidden single method is similar to the naked single method but the way to find the way to find the empty square is different. When there is only one square in the row, column or box that can take a certain number, then the square must take that number. For example in below figure , we can see that both row2 and row3 contain the digit 9 so according to the rules, row1 must also hold number 9 (in the square 123def). In the right side of figure below, number 9 is inserted by using the hidden singles method.

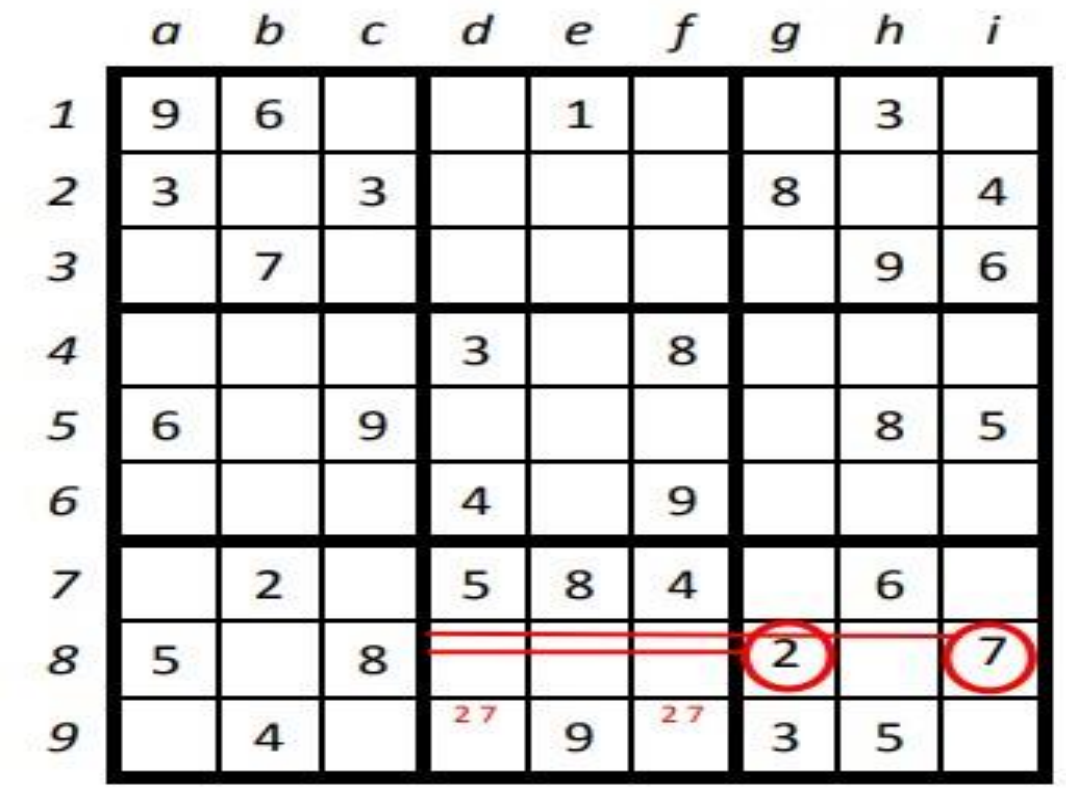


Hidden Single Technique

The figure show how the hidden single technique can be used

d.Locked candidate: Tom Davis has stated that it is possible to find a box where the only place for a candidate would be a row or column within that box . If a candidate belongs to a row or column then we can remove this candidate as a possible one with other boxes that the row (or column) connected with them

e. Naked Pairs, Triplets: These techniques are very similar to the naked single technique, but in this method we find the same two candidates in two squares. By using this information we can find a possible candidate to other squares. For example in below figure , squares 9d and 9f can only contain values 2 and 7. By having this knowledge, it is obvious that square 9d and 9f cannot contain 1 or 6 so those candidates are removed. The only candidates are 2 and 7 in squares 9d and 9f.



Naked Pair Technique

C. Brute Force Algorithm

The second algorithm that is examined in this work is Brute force algorithm. Usually, the brute force algorithm can be applied to any possible algorithm. For example when finding password, the algorithm generates any possible password until the right one is found. In this case the algorithm goes through every empty square and places a valid digit in that square. If no valid number is found the algorithm comes back to the previous square and change the value in that square. The process is repeated until the board is filled with numbers from 1 to 9. The advantage of the brute force algorithm is that the algorithm can guarantee a solution to any puzzles since it generates all possible answers until the right answer is found if the puzzles are valid . Additionally, the running time can be unrelated to level of difficulty, because the algorithm searches for every possible solution. In order to compare the pencil-and-paper algorithm with the brute force algorithm a (pre-implemented) brute force algorithm has been used during testing .

The brute-force algorithm is a systematic, exhaustive search approach employed to solve Sudoku puzzles. It explores potential solutions by iteratively trying all possible combinations until a valid solution is found or all possibilities are exhausted. This algorithm is characterized by its simplicity and reliability but may become computationally expensive for larger or more complex puzzles

Completeness: The brute-force algorithm is complete and guarantees finding a solution if one exists.

Time Complexity: The time complexity can be high, especially for complex puzzles, making optimization crucial for practical use.

Memory Usage: Recursive nature may lead to a high memory footprint, especially for large puzzles.

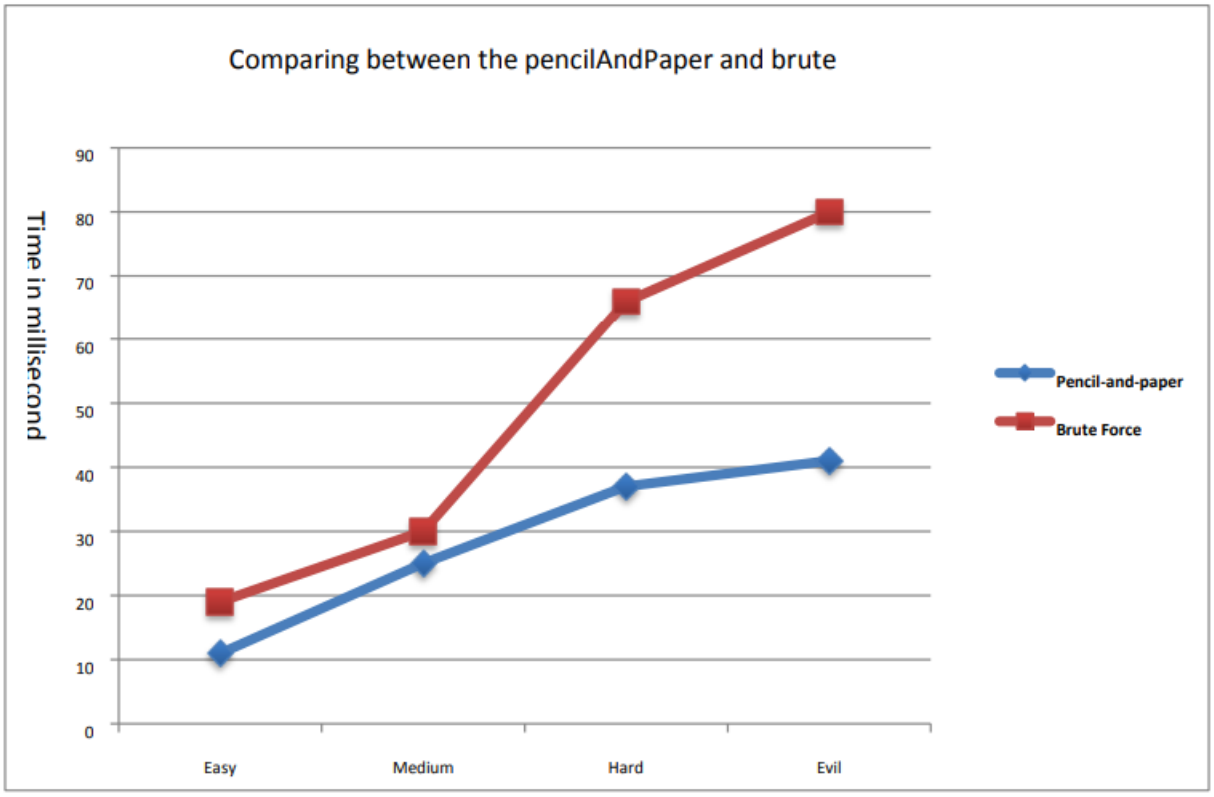
Parallelization: Parallelization is possible by exploring multiple branches simultaneously, improving performance.

D. Pencil-and-paper Algorithm vs Brute Force Algorithm

Criteria	Paper-and-Pen Algorithm	Brute-Force Algorithm
Approach	Human-driven logical deduction	Systematic trial and error
Complexity	Less complex	More complex
Speed	Depends on the solver's skill	Can be slow, especially for larger puzzles
Deterministic	Yes, based on human reasoning	Yes, but relies on exhaustive search
Optimization	Human intuition may optimize solving path	Limited optimization, explores all possibilities
Memory Usage	Low	Can be memory-intensive, especially for recursive backtracking
Suitability for Humans	Intuitive, preferred by some players	Not practical for solving large or complex puzzles
Suitability for Machines	Difficult to automate	Easily automated, suitable for implementation in algorithms
Real-Time Interaction	Yes, immediate feedback	No, requires completion of entire search space
Practicality	Limited for complex puzzles	General-purpose, applicable to a wide range of Sudoku problems
Parallelization	Not applicable	Possible, especially with parallel or distributed systems

This table provides a concise comparison between the two approaches, highlighting their characteristics in terms of approach, complexity, speed, determinism, optimization, memory usage, suitability for humans and machines, real-time interaction, practicality, and parallelization.

The proposed algorithm has proved that is able to solve Sudoku puzzle with any levels of difficulty. We have assumed to have four levels of difficulty during testing. These levels are; easy, medium, hard and evil (challenging). This algorithm is able to solve the easy and medium puzzle without using backtracking method (less than 20 ms). In order to solve the puzzles with more difficult level such as hard or evil the algorithm uses the backtracking method as well. During testing we have noticed that the given algorithm performs better than the brute force algorithm in the term of the runtime (the time the algorithm takes to be executed). The diagram below shows the differences between these two algorithms. Note that the puzzles, which are used in the testing, are taken from a valid webpage . The webpage generates Sudoku puzzles with different ratings.



A comparison between the pencil-and-paper algorithm and brute force algorithm

This diagram depicts the differences between the pencil-and-paper algorithm and the brute force algorithm based on how long it takes to solve the puzzles by a computer. The penciland-paper algorithm solves the puzzle quicker than the brute force algorithm. The given data is based on the averaging of the computing time for several puzzles that have been tested with the same difficulty levels such as easy, medium, hard, and evil respectively. For instance the time obtained for the easy level is the result of averaging several computing times with easy level. In the given diagram the vertical axis represents running time of the puzzles in milliseconds and the horizontal axis the difficulty levels.

II. Sudoku Rules for programming

Understanding the rules of Sudoku is fundamental to solving the puzzle and serves as the cornerstone of the backtracking algorithm's application.

A. Rules

Reiterating the key rules:

- A 9x9 grid is divided into 3x3 subgrids.
- Digits 1 to 9 must fill the grid.
- No repetition is allowed in rows, columns, or subgrids.

B. Uniqueness and Solvability

Highlighting the importance of the puzzle's unique solution and the logical steps required for a human to solve it. Emphasizing the role of backtracking in systematically exploring potential solutions.

III. Introduction to Backtracking

Backtracking, a fundamental algorithmic technique, provides an elegant solution to constraint satisfaction problems like Sudoku. As we delve into its nuances, we uncover its applications beyond puzzles, offering a versatile approach to problem-solving.

A. Algorithmic Overview

Breaking down the backtracking algorithm into its core components:

- **Exploration:** Systematically exploring possible solutions.
- **Constraint Satisfaction:** Ensuring solutions adhere to specified constraints.
- **Backtracking Mechanism:** Efficiently navigating through the solution space.

●Pseudo code:

Here's a pseudocode explanation for the main functions:

1.print_grid(grid):

- Iterate through each row of the grid.
- For each row, convert each element to a string and join them with a space.
- Print the joined string for each row.

2.find_empty_location(grid, empty_loc):

- Iterate through each cell in the grid.
- If a cell contains 0 (empty), update the empty_loc list with the current row and column.
- Return True if an empty cell is found; otherwise, return False.

3.used_in_row(grid, row, num):

- Check if the given number num is not present in the specified row of the Sudoku grid.

4.used_in_col(grid, col, num):

- Check if the given number num is not present in the specified column of the Sudoku grid.

5.used_in_box(grid, row, col, num):

- Determine the starting row and column for the 3x3 box containing the specified cell.
- Check if the given number num is not present in the 3x3 box.

6.is_safe(grid, row, col, num):

- Check if the number num can be safely placed in the specified cell.
- Verify that the number is not present in the current row, column, and 3x3 box.

7.solve_sudoku(grid):

- Initialize empty_loc to store the row and column of an empty cell.
- If no empty cell is found, return True (the Sudoku is solved).
- Get the row and column of the empty cell.
- Try placing numbers from 1 to 9 in the empty cell.
- If placing a number is safe, recursively call solve_sudoku on the updated grid.
- If the recursive call returns True, the Sudoku is solved; otherwise, backtrack by setting the cell value back to 0.
- If no number can be placed in the current empty cell, return False.

8.Example Sudoku grid (sudoku_grid):

- A 9x9 grid representing a Sudoku puzzle. 0 represents an empty cell.

9.Solving the Sudoku:

- Call solve_sudoku with the provided Sudoku grid.
- If a solution exists, print the solved Sudoku grid using print_grid.
- If no solution exists, print "No solution exists."

Methods involve in solving sudoku

Method	Description
print_grid(grid)	Prints the Sudoku grid, converting each element to a string and joining them with spaces for each row.
find_empty_location(grid, empty_loc)	Finds the first empty (0) cell in the Sudoku grid and updates the **empty_loc** list with its row and column. Returns **True** if an empty cell is found, **False** otherwise.
used_in_row(grid, row, num)	Checks if the given number **num** is not present in the specified row of the Sudoku grid.
used_in_col(grid, col, num)	Checks if the given number **num** is not present in the specified column of the Sudoku grid.
used_in_box(grid, row, col, num)	Checks if the given number **num** is not present in the 3x3 box containing the specified cell in the Sudoku grid.
is_safe(grid, row, col, num)	Checks if placing the given number **num** in the specified cell is safe (not present in the row, column, and 3x3 box).
solve_sudoku(grid)	Solves the Sudoku puzzle using the backtracking algorithm. Returns **True** if a solution exists, **False** otherwise. Backtracks when necessary.

B. Backtracking in Real-world Applications

Beyond Sudoku, examining instances where backtracking finds utility, such as in resource allocation, scheduling, and network routing problems. Highlighting its adaptability and efficacy in various domains.

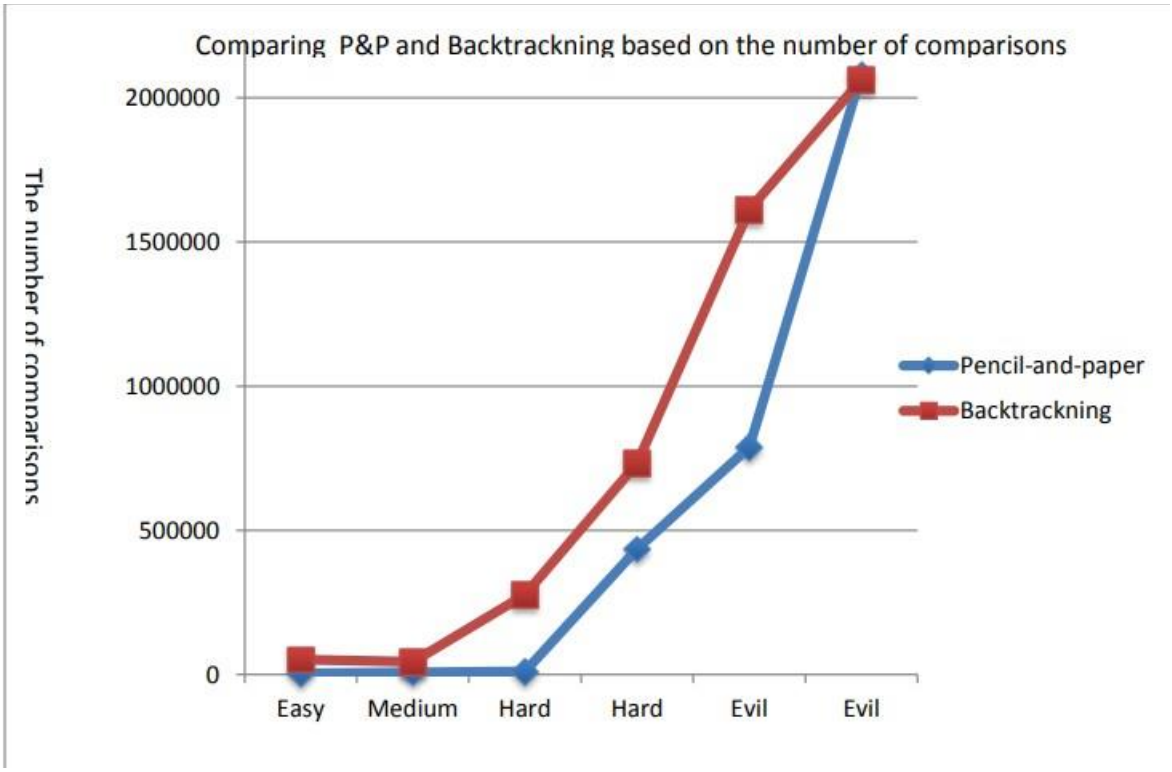
C. Backtracking Algorithm vs Pencil-and-paper Algorithm

Generally, the backtracking method, which is similar to the brute force algorithm, can solve the puzzles quicker than the pencil-and-paper algorithm. The question now is why should we use backtracking and human methods together? There are three reasons to do so. Firstly, the purpose of this work is to implement an algorithm applying human strategies. Secondly, human players also use the backtracking method when they get stuck. It means that players check different alternatives and place the numbers in the empty squares by guessing when there are no options left.

Pencil and Pen algorithm vs Backtracking algorithm

Criteria	Pencil and Pen Algorithm	Backtracking Algorithm
Basic Idea	Eliminates possibilities using pencil marks	Systematically explores possible solutions
Approach	Human-like, mimics how humans solve Sudoku	Systematic search and trial-and-error
Speed	Generally faster for simpler puzzles	Generally slower for simpler puzzles
Complexity	Less complex, relies on logical deductions	More complex, involves recursion and trials
Space Complexity	Lower space complexity	Higher space complexity
Memory Usage	Less memory usage	More memory usage
User Interaction	Mimics human solving with pencil marks	Directly solves without user interaction
Applicability	Well-suited for easier puzzles and human-like solving	General-purpose, applicable to all puzzles
Efficiency for Sudoku	May struggle with very difficult puzzles	Efficient, can solve any solvable Sudoku
Parallelization	Limited parallelization opportunities	Potential for parallelization in some cases

Finally, employing the human strategies make the algorithm more efficient based on the number of comparisons. In other words, the naked single method fills the empty squares by performing fewer comparisons in short time as it uses a better technique. However, the backtracking method runs more number of analytical circulation while solving the puzzles resulting consumption of memory space. This can clearly be shown in the diagram. The result shows that the number of comparisons in the backtracking method is higher than when the human strategies and backtracking methods are used together.



Comparing the pencil-and-paper algorithm and the backtracking algorithm

IV. Data Structures Used

Effective data structures form the backbone of any algorithm. In Sudoku solving, the choice of data structures profoundly influences the efficiency of the backtracking process.

A. Grid Representation

Detailing the 2D array representation of the Sudoku grid and its implications for quick access and manipulation.

B. Set or Array for Candidates

Exploring the use of sets or arrays to store potential candidate numbers for each empty cell, streamlining the decision-making process during backtracking.

C. Stack or Recursive Call Stack

Discussing the role of a stack or the call stack in managing the recursive nature of the backtracking algorithm, ensuring a systematic exploration of possibilities.

V. Approach

The step-by-step breakdown of the backtracking algorithm is crucial for grasping its practical implementation in Sudoku solving.

A. Empty Cell Identification

A detailed discussion on identifying and managing empty cells, emphasizing the importance of this initial step in the backtracking process.

B. Number Assignment

Delving into the strategy of assigning numbers to empty cells, discussing heuristic approaches and their impact on the overall efficiency.

C. Check Validity

Examining the checks in place to ensure that the assigned number maintains the integrity of the Sudoku rules. Emphasizing the importance of these validations in pruning the solution space.

D. Backtrack Mechanism

Detailing the algorithm's ability to backtrack when an invalid assignment is encountered, effectively 'undoing' decisions and exploring alternative paths.

E. Optimization Techniques

Introducing optimization strategies to enhance the algorithm's efficiency, such as constraint propagation and forward checking.

VI. Backtracking Algorithm vs Pencil-and-paper Algorithm vs Brute-force

Algorithm	Description	Complexity	Pros	Cons
Pencil and Pen Algorithm	This algorithm involves using "pencil marks" to eliminate candidates and then using "pen marks" for certain placements.	Not well-defined	Can be more human-like in approach	May not guarantee a solution
Backtracking Algorithm	A recursive algorithm that systematically explores possible solutions and backtracks when it reaches an invalid state.	Exponential	Guarantees a solution	Can be slow for difficult puzzles
Brute Force Algorithm	A straightforward approach where all possible combinations are tried until a valid solution is found.	Exponential	-Simple to implement	Highly inefficient for most puzzles

The time complexity for Backtracking and Brute Force algorithms is typically exponential in the worst case, but the actual performance depends on the specific puzzle and the implementation details. The Pencil and Pen Algorithm's complexity is not well-defined as it is more of a concept used by human players.Keep in mind that in practice, efficient Sudoku solvers often use a combination of techniques and heuristics to improve performance.

VII. Conclusion

Summarizing the key insights gained from the exploration of Sudoku solving using the backtracking algorithm, reflecting on the significance of algorithmic approaches in solving real-world problems.

A. Algorithmic Thinking

Highlighting the importance of algorithmic thinking in problem-solving and its applicability beyond Sudoku puzzles.

B. Community and Collaboration

Encouraging collaboration within the community, sharing insights, and contributing to the collective knowledge pool.

1.3 Technology Stack

I. Programming Language (Python)

Python was chosen as the primary programming language for its simplicity, readability, and versatility. It offers an extensive set of libraries and frameworks, making it well-suited for implementing data structures and algorithms.

II. Project-Specific Technologies (Sudoku Solver Project)

A. Data Structures:

a. Arrays: Used for efficient representation and manipulation of the Sudoku grid.

b. Sets: Employed for storing candidate numbers for each empty cell, facilitating decision-making during backtracking.

c. Stacks or Recursive Call Stack: Managed the recursive nature of the backtracking algorithm, ensuring systematic exploration of possibilities.

B. Algorithm (Backtracking Algorithm):

The project focused on implementing the backtracking algorithm to solve Sudoku puzzles systematically.

III. Development Environment

A. Integrated Development Environment (IDE):

The choice of IDE depends on personal preference, but popular options include:

- **PyCharm:** A powerful Python IDE with features for code completion, debugging, and project navigation.
- **Jupyter Notebooks:** Useful for interactive development and testing, especially when exploring algorithms step by step.
- **IDLE:** IDLE is an integrated development environment for Python, providing a convenient environment for writing, testing, and debugging Python code.

IV. Version Control

A. Git: Git was used for version control, enabling collaborative development, tracking changes, and managing project history.

V. Documentation

A. Markdown:

Markdown was employed for creating structured and easily readable documentation. It supports text formatting and can be converted to various formats.

VI. Collaboration and Communication

A.Communication Tools:

- **Email:** Used for formal communication, submitting reports, and receiving feedback.
- **Instant Messaging:** Platforms like Slack or Microsoft Teams may have been used for real-time communication within the team.

VII. Presentation

A. Presentation Tools:

- **Microsoft PowerPoint or Google Slides**: Likely used for creating presentations summarizing the internship experience, project details, and learnings.

VIII. Project Management

A. Project Management Tools:

- Trello, Asana, or Jira: These tools help in organizing tasks, tracking progress, and managing the overall project workflow.

IX. Cloud Services

A. Cloud Platforms:

- **GitHub or GitLab:** Used for hosting the project repository, enabling version control and collaboration.

B. Google Drive or Microsoft OneDrive:

Potentially used for storing and sharing project-related documents.

X. Web Technologies (if applicable)

A. Web Frameworks:

If the project involved a web-based component, frameworks like Flask or Django might have been used for backend development.

XI. Security (if applicable)

A. Security Tools:

- **Python Security Libraries:** Utilized to ensure secure coding practices, prevent vulnerabilities, and protect against common security threats.

XII. Testing

A. Testing Frameworks:

- **PyTest**: If automated testing was implemented, PyTest might have been used for unit testing.

XIII. Database (if applicable)

A. Database Systems:

- **SQLite, MySQL, or PostgreSQL:** Depending on project requirements, a database system might have been integrated.

XIV. Continuous Integration/Continuous Deployment (CI/CD)

A. CI/CD Tools:

- **Jenkins, Travis CI, or GitLab CI:** These tools automate the testing and deployment process, ensuring a streamlined development pipeline.

This technology stack reflects the tools and technologies that might have been employed during the internship, particularly for the Sudoku Solver Project. The specific tools and versions may vary based on individual preferences and project requirements.

CHAPTER 2

SUMMARY OF EXPERINCE

During my internship at YBI Foundation, I had the opportunity to delve into the realm of emerging technologies, with a specific focus on data structures and their implementation using the Python programming language. The experience provided valuable insights into both theoretical concepts and practical applications, fostering a holistic understanding of the subject matter.

A. Learning Journey

The internship commenced with a comprehensive exploration of fundamental data structures, including arrays, linked lists, stacks, queues, trees, and graphs. The theoretical foundation laid the groundwork for practical implementation, enhancing my ability to design and optimize algorithms.

B. Python Proficiency

One of the primary objectives was to leverage Python's simplicity and versatility for implementing data structures effectively. The hands-on experience in coding not only improved my Python proficiency but also instilled confidence in solving real-world problems using the language's rich set of libraries.

C. Project Focus: Sudoku Solving

The core project involved the implementation of a Sudoku solver using the backtracking algorithm. This project encapsulated the application of various data structures, emphasizing the role of arrays, sets, and recursive call stacks. The experience of building a solution to a well-known problem solidified my understanding of algorithmic thinking and problem-solving strategies**.**

D. Optimization Techniques

A crucial aspect of the internship was the exploration of optimization techniques. Understanding how to judiciously select and use data structures contributed to writing efficient and scalable code. The emphasis on constraint propagation and forward checking provided insights into enhancing program efficiency.

E. Collaborative Learning Environment

The internship fostered a collaborative learning environment, encouraging teamwork on practical projects. Engaging with peers, sharing insights, and collectively tackling coding challenges enhanced my ability to work collaboratively, mirroring real-world software development scenarios.

F. Professional Guidance

Under the esteemed supervision of Dr. G. Sunitha, a seasoned professor in the Department of Computer Science and Engineering, I received valuable guidance and mentorship. The support extended by the faculty contributed significantly to my learning experience.

CHAPTER 3

REFLECTION ON LEARNING

Reflecting on the internship, I recognize the substantial growth in my understanding of data structures and algorithmic problem-solving. The hands-on projects, especially the Sudoku solver, provided a tangible application of theoretical knowledge.

A. Key Takeaways

Practical Implementation: The internship emphasized the importance of practical implementation in reinforcing theoretical concepts. Building solutions to real-world problems deepened my understanding of data structures.

- **Python as a Tool:** The choice of Python as the primary programming language showcased its effectiveness in implementing data structures. The language's simplicity and readability proved instrumental in rapid development.
- **Optimization Mindset:** Learning optimization techniques underscored the significance of making informed decisions about data structures. Evaluating trade-offs and choosing the right structures contributed to efficient code.

B. Future Applications

The skills acquired during the internship lay a strong foundation for future endeavors. The ability to apply data structures in problem-solving scenarios and the proficiency gained in Python programming are transferable skills applicable across diverse domains.

C. Recommendations

I recommend incorporating more industry-oriented projects in future internships, providing exposure to real-world challenges. Additionally, integrating coding competitions or challenges could further enhance problem-solving skills and competitiveness among interns.

CONCLUSION

In conclusion, my internship at YBI Foundation has been an enriching and transformative experience, leaving an indelible mark on my journey in the realm of computer science. Over the course of the internship, I delved into the intricate world of data structures, gaining not only theoretical knowledge but also practical insights into their implementation. The hands-on projects, with a particular focus on the Sudoku Solver utilizing the backtracking algorithm, served as a real-world application of the concepts I acquired.

The internship became a crucible for refining my Python programming skills, leveraging the language's simplicity and flexibility for effective implementation. Through various coding exercises and projects, I not only strengthened my proficiency in Python but also developed a nuanced understanding of its capabilities in solving complex problems.

The collaborative learning environment at YBI Foundation fostered a sense of camaraderie among interns. Engaging with peers in team projects, sharing insights, and collectively solving coding challenges enhanced not only my technical abilities but also my interpersonal skills. The synergy created within the team mirrored the collaborative nature of the software development industry, preparing me for future professional endeavours.

The professional guidance provided by Dr. G. Sunitha, a seasoned professor in the Department of Computer Science and Engineering, played a pivotal role in shaping my learning experience. The mentorship offered valuable insights, bridging the gap between theoretical concepts and their practical applications. This guidance was instrumental in navigating through complex problem-solving scenarios and understanding the nuances of algorithmic thinking.

This internship has not only expanded my academic horizons but has also instilled a problem-solving mindset that extends beyond the realm of data structures and Python programming. The emphasis on optimization techniques, such as constraint propagation and forward checking, has equipped me with a strategic approach to enhance the efficiency of algorithms, a skill applicable across various domains in computer science.

In essence, the internship at YBI Foundation has been a holistic and well-rounded journey. It has deepened my knowledge base, honed my technical skills, and provided a glimpse into the collaborative and dynamic world of computer science. As I conclude this transformative experience, I carry forward not just a certificate of completion but a comprehensive set of skills, perspectives, and a passion for continuous learning in the ever-evolving field of technology.