Texas A&M University - Commerce

Department of Computer Science

Evaluating the Proficiency of ChatGPT in Undergraduate Data Structures and Algorithms: An Analysis of Standardized Test Performance

Mokshith Ramendra Yaganti

Supervisor: Derek Harter, Ph.D.

A report submitted in partial fulfilment of the requirements of
Texas A&M University - Commerce for the degree of
Master of Science in *Computer Science*

May 2, 2024

Declaration

I, Mokshith ramendra Yaganti, of the Department of Computer Science, Texas A&M University - Commerce, confirm that this is my own work and figures, tables, equations, code snippets, artworks, and illustrations in this report are original and have not been taken from any other person's work, except where the works of others have been explicitly acknowledged, quoted, and referenced. I understand that if failing to do so will be considered a case of plagiarism. Plagiarism is a form of academic misconduct and will be penalised accordingly.

I give consent to a copy of my report being shared with future students as an exemplar.

I give consent for my work to be made available more widely to members of TAMUC and public with interest in teaching, learning and research.

Mokshith ramendra Yaganti
May 2, 2024

Abstract

The realm of Artificial Intelligence (AI) is undergoing a significant transformation, primarily driven by the advancements in Large Language Models (LLMs). Among these, ChatGPT has emerged as a standout model, acclaimed for its exceptional ability in conducting multi-turn conversations and showcasing coding proficiency in various programming languages. This study deals with the investigation of the performance of ChatGPT and the impact of prompt engineering, on its effectiveness in solving standardized undergraduate-level data structures and algorithms problems. A novel aspect of this research is the focus on automating the entire evaluation pipeline, including prompt fine-tuning, generating responses from ChatGPT, and systematically testing these responses against a curated set of standard questions. This automation not only streamlines the assessment process but also sets a precedent for analyzing other LLMs in a similar fashion.

The methodology centers on the development and application of tailored prompts designed to maximize ChatGPT's performance in solving complex programming challenges. The study meticulously curates a diverse collection of data structures and algorithms questions, representative of undergraduate coursework. ChatGPT's responses to these prompts are then automatically processed and evaluated against multiple test cases to determine their correctness and efficacy.

Key findings of this research will illuminate the potential of prompt engineering as a crucial factor in enhancing the performance of LLMs in technical domains. The outcomes are expected to provide valuable insights into the capabilities and limitations of ChatGPT in the context of algorithmic problem-solving. Furthermore, the study's automated approach promises scalability and reproducibility, offering a robust framework for future research in LLM performance analysis across various disciplines.

Keywords: ChatGPT, Large Language Models, Prompt Engineering, Data Structures, Algorithms.

Contents

List of Figures

List of Tables

List of Abbreviations

SMPCS	School of Mathematical, Physical and Computational Sciences

Chapter 1

Introduction

1.1 Background

The proficiency of Large Language Models (LLMs) like ChatGPT in generating working code is intrinsically linked to their understanding and application of data structures and algorithms (DSA). These fundamental components form the backbone of efficient and effective software development. DSA are critical in determining how data is organized, stored, and manipulated within a program, impacting everything from the execution speed to resource utilization. The ability of an AI model to adeptly handle these aspects is indicative of its depth in coding knowledge and its applicability in real-world software development scenarios.

In this context, the capabilities of ChatGPT in DSA are particularly noteworthy. This model has demonstrated a remarkable ability to not only understand and implement standard data structures and algorithms but also to apply them creatively to solve complex problems. The implication of this proficiency is significant; it suggests that LLMs like ChatGPT can be invaluable tools in the software development process, assisting in everything from initial problem analysis to the formulation of efficient algorithmic solutions.

Furthermore, the application of DSA in code generation by LLMs also opens up possibilities for more advanced software development applications. For instance, an AI model proficient in DSA can potentially assist in optimizing existing codebases, refactoring inefficient structures, or even suggesting algorithmic improvements.

In the current landscape of AI-driven code generation, particularly with models like ChatGPT, there is a noticeable challenge in consistently generating code that is both efficient and correct. The generation of correct code is fundamental, as errors in code can lead to significant issues in software development, ranging from minor bugs to critical system failures. While ChatGPT exhibits a strong capacity for code generation especially in DSA, as evidenced in the work by Arefin et al. (2023), its performance can vary, especially in terms of efficiency and adherence to best practices in DSA. This variability underscores the necessity for more refined techniques in interacting with these models to elicit the highest quality of code output.

Prompt engineering emerges as a crucial element in this context. It plays a critical role in harnessing their full potential, especially in code generation. It involves meticulously crafting input prompts to effectively communicate the requirements and constraints of a given problem to the AI model. This practice is particularly essential in programming scenarios where the quality of output is directly influenced by how the problem is presented to the model, i.e., the clarity

and specificity of instructions can significantly influence the accuracy and utility of the generated code. A well-engineered prompt can lead to solutions that are not only syntactically and logically correct but also optimized in terms of performance and resource management. The significance of prompt design in obtaining optimal results from generative tasks has been highlighted in the research by Chen et al. (2021) on GPT models.

The implications of a system proficient in accurately generating working, logical code are far-reaching. In the future, such capabilities could revolutionize software development, enabling faster deployment of robust and sophisticated applications. Furthermore, as noted by Chen et al. (2021) in their analysis of code generation models, these advancements could democratize programming, allowing individuals with limited coding expertise to develop software solutions through intuitive, natural language instructions. This could potentially lead to a surge in innovation, as barriers to software creation are lowered, and a broader range of perspectives are brought into the technology development process.

The current research aims to delve deeper into ChatGPT's capabilities in DSA. It aims to assess and quantify the impact of prompt engineering on ChatGPT's performance in generating algorithmic solutions. By evaluating the model's performance in this domain, the study seeks to shed light on the extent to which ChatGPT can accurately and effectively generate code solutions that are not just correct in their logic, but also optimal in their use of data structures and algorithms. This exploration is pivotal, as it directly relates to the practical utility of LLMs in software engineering, a field where efficiency and optimization are paramount.

1.2 Problem statement

The central problem this study addresses is the assessment of the impact of prompt engineering on the performance of ChatGPT, specifically in solving undergraduate-level data structures and algorithms (DSA) questions. The effectiveness of Large Language Models (LLMs) like ChatGPT in code generation has been increasingly recognized. However, their ability to consistently produce efficient and correct solutions in the context of complex DSA problems remains an area requiring deeper exploration. This research hypothesizes that through strategic prompt engineering, the proficiency of ChatGPT in generating accurate and optimized solutions to DSA challenges can be significantly enhanced.

Prompt engineering, in this context, refers to the deliberate design and structuring of input prompts to guide ChatGPT in understanding and effectively responding to the intricacies of DSA problems. The hypothesis is grounded in the premise that the manner in which a problem is presented to an LLM can profoundly influence the quality of the generated solution. This hypothesis aligns with the findings of Radford et al. (2019), who highlighted the importance of prompt design in achieving desired outcomes from generative AI models. The study aims to empirically test this hypothesis by systematically varying the prompts used to interact with ChatGPT and evaluating the resulting code's correctness, efficiency, and adherence to DSA best practices.

The significance of this investigation lies not only in its potential to enhance the understanding of prompt engineering as a tool for optimizing LLM performance but also in its broader implications for the field of AI-driven software development. By establishing a clear correlation between prompt engineering and the quality of ChatGPT-generated code, the study seeks to contribute to the development of more effective methodologies for leveraging LLMs in technical and educational

settings.

1.3 Aims and objectives

The overarching aim of this project is to conduct a comprehensive evaluation of ChatGPT's capabilities in solving data structures and algorithms problems typically encountered at the undergraduate level. This aim encompasses several specific objectives:

Development of Tailored Prompts: One of the primary objectives is to develop tailored prompts that enhance ChatGPT's problem-solving skills in data structures and algorithms. This involves designing prompts that not only accurately describe the problem but also guide the model towards optimal problem-solving approaches. The effectiveness of these prompts will be gauged by their ability to elicit accurate, efficient, and sophisticated solutions from ChatGPT.

Curating a Comprehensive Standard Test Set: The project will involve curating a comprehensive set of standard undergraduate-level questions in data structures and algorithms. The selected test set for this research includes a combination of Easy, Medium and Hard questions which reflect a standard DSA question set an undergraduate student might encounter in a real world scenario.

Automation of the Evaluation Process: Automating the evaluation of ChatGPT's responses is another critical objective. This will be achieved by integrating ChatGPT with the LeetCode API (graphql), allowing for an efficient and systematic assessment of the solutions' correctness and practical applicability. Automation will ensure consistency in evaluation and enable the handling of a large volume of test cases.

Comparison of Effectiveness: The project aims to compare the effectiveness of ChatGPT's responses with and without prompt engineering. This comparative analysis will help ascertain the impact of prompt engineering on the model's performance. It will involve analyzing the solutions generated by ChatGPT under different prompting conditions to evaluate how variations in prompt design influence the accuracy and quality of the generated code.

Through these objectives, the project seeks to provide a detailed analysis of how effectively ChatGPT can be utilized in the domain of data structures and algorithms, and the extent to which prompt engineering can enhance its performance. The results are expected to offer valuable insights into the practical applications of LLMs in technical education and software development.

1.4 Solution approach

The methodology of this project is structured around a blend of prompt engineering and automated evaluation techniques, designed to rigorously test ChatGPT's proficiency in solving data structures and algorithms problems.

By employing these methodical and automated approaches outlined below, the study aims to provide a detailed analysis of ChatGPT's problem-solving abilities in the context of data structures and algorithms. The findings are expected to offer significant insights into the potential of LLMs in technical education and software development, particularly in the realm of algorithmic thinking and coding proficiency.

1.4.1 Automated System Development and Integration

A key component of our methodology involves developing an automated system that interfaces with both the ChatGPT and LeetCode APIs. This system will automatically fetch questions from LeetCode, using graphql API behind leetcode.com. These questions, representative of common undergraduate-level data structures and algorithms challenges, form the basis of our evaluation set.

1.4.2 Solution Generation and Submission

Once a question is fetched, it is passed to the ChatGPT API to generate a solution. This process is repeated multiple times for each question, with each iteration using a different set of prompts. These prompts are strategically designed to guide ChatGPT towards generating more effective solutions. Each generated solution is then submitted back to LeetCode through its API, enabling us to assess its correctness and efficiency.

1.4.3 Evaluation Metrics

The primary metrics for evaluating each solution are the number of test cases passed and the run time. The number of test cases passed is a direct indicator of the solution's correctness, reflecting how well the code meets the problem's requirements. The run time, on the other hand, provides insight into the efficiency and optimization of the solution. Together, these metrics are critical for evaluating the effectiveness of different prompts and the overall capability of ChatGPT in solving complex DSA questions.

1.4.4 Comprehensive Testing Across Standard Test Questions

This entire process, from question retrieval to solution evaluation, is systematically repeated for all the standard test questions outlined in the Aims & Objectives section. This comprehensive approach ensures a thorough assessment of ChatGPT's capabilities and the impact of various prompt engineering strategies on its performance.

1.5 Summary of Contributions and Achievements

This project has achieved significant milestones in evaluating the capabilities of ChatGPT in solving complex data structures and algorithms (DSA) problems through an automated testing framework. Major contributions of this work include:

- **Development of an Automated System:** A fully automated pipeline was created for fetching coding problems, generating solutions using ChatGPT, and evaluating these solutions against standardized test cases. This system is scalable and can be adapted for broader AI testing applications.
- **Role-Based Prompt Engineering:** This work extensively explored how different prompts influence the performance of ChatGPT. Through careful design of role-based prompts, it was possible to direct the AI to approach problems with varying strategies, significantly affecting the quality of the solutions generated.

- **Comprehensive Testing and Analysis:** A vast array of DSA problems was used to test the model, providing a deep insight into its strengths and weaknesses across different topics. The findings reveal that while ChatGPT excels in areas requiring mathematical reasoning, it struggles with tasks that need complex data structure manipulation.
- **Identification of Key Patterns:** The project identified critical patterns in errors and solution quality, which can help refine the AI model's training and prompt engineering for improved performance.

The implications of these results are vast, providing a foundation for future research in AI-driven coding solutions and enhancing the understanding of how language models can be effectively utilized in programming education and professional software development.

1.6 Organization of the Report

This report is organized into seven main chapters, each dedicated to a different aspect of the research, ensuring a logical flow and comprehensive understanding of the project.

- **Chapter 1: Introduction** - This chapter sets the stage for the report, outlining the research problem, objectives, and the scope of the study.
- **Chapter 2: Literature Review** - Details the existing research on AI capabilities in coding, specifically focusing on the performance of language models like ChatGPT in technical domains.
- **Chapter 3: Methodology** - Describes the methods used for data collection, prompt engineering, solution generation and submission, and the automated system development and integration.
- **Chapter 4: Results** - Presents the findings from the extensive testing of ChatGPT, analyzing its performance across various prompts and topics.
- **Chapter 5: Discussion and Analysis** - Discusses the implications of the results, the effectiveness of different prompts, and the model's performance on various DSA topics.
- **Chapter 6: Conclusions and Future Work** - Summarizes the key findings and discusses potential areas for future research.
- **Chapter 7: Reflection** - Provides a personal account of the research experience, reflecting on the learning and challenges encountered during the project.

Appendices and additional supporting material can be found at the end of the report, providing further details on data and methodologies used throughout the research.

Chapter 2

Literature Review

In the rapidly evolving field of artificial intelligence, the exploration of Large Language Models (LLMs) like ChatGPT has become a focal point for researchers aiming to harness their capabilities for both coding and non-coding applications. Our study situates itself within this dynamic research landscape, specifically investigating ChatGPT's proficiency in data structures and algorithms (DSA)—a core component of computer science that underpins efficient software development. This literature review synthesizes pivotal findings from leading studies, setting the stage for our inquiry into how strategic prompt engineering can refine ChatGPT's DSA problem-solving abilities.

2.0.1 ChatGPT's Coding Capabilities

CRUD Operations and Programming Assistance

Noever and McKee (2023)'s seminal work offers an insightful evaluation of ChatGPT's ability to execute CRUD operations across several well-known datasets, revealing the model's adeptness at mimicking Python interpreter functionalities. This capacity for autonomous code generation and execution marks a significant stride in AI's integration into software development, particularly in handling structured data—a critical aspect our research aims to build upon by examining ChatGPT's effectiveness in more complex coding scenarios inherent in DSA challenges.

Biswas (2023) extend the discussion on ChatGPT's utility as a programming assistant, underscoring its potential to streamline code completion, debugging, and refactoring tasks. This study illuminates the model's capacity to alleviate the manual burden associated with these activities, emphasizing its role as a facilitator of more efficient coding practices. Our research echoes this theme, delving into how ChatGPT can be further optimized through prompt engineering to tackle the nuanced demands of DSA problem-solving.

Advanced Coding Skills

The work of Bubeck et al. (2023) aligns closely with our methodological approach, challenging ChatGPT to develop Python functions using diverse problem sets. Their findings not only highlight GPT-4's superior performance but also identify critical areas for improvement when dealing with intricate coding tasks. This insight is particularly relevant to our study, which seeks to elucidate

how varying prompt designs might enhance ChatGPT's ability to generate more accurate and efficient DSA solutions.

Tian et al. (2023)'s comprehensive analysis further details ChatGPT's capabilities in code translation, correction, and comprehension. While acknowledging the model's sophisticated coding proficiency, the study also points to its occasional inaccuracies and inefficiencies—challenges our research addresses by investigating the potential of prompt engineering to mitigate such issues in the context of DSA problems.

The innovative study by Feng et al. (2023) utilizes a crowdsourcing approach to assess ChatGPT's code generation capabilities. By analyzing public discourse on social media, they provide a nuanced understanding of common programming languages, usage scenarios, and prevalent errors in code snippets generated by ChatGPT. This exploration of public perception offers a backdrop against which our research examines the specificity and clarity of prompts in refining ChatGPT's output for DSA tasks.

2.0.2 ChatGPT's Performance in Non-Coding Domains

Medical and Mathematical Skills

The research conducted by Gilson et al. (2023) evaluates ChatGPT's capacity to respond to medical licensing exam questions, revealing its proficiency in recall-based queries but limitations in complex reasoning tasks. This delineation between ChatGPT's strengths and weaknesses in processing medical data provides a parallel to our study's focus on the model's problem-solving strategies within DSA, particularly how prompt engineering can address gaps in reasoning and technical accuracy.

Studies by Frieder et al. (2023) delve into ChatGPT's mathematical abilities, highlighting its potential for creative reasoning alongside notable deficiencies in critical thinking and precision. These findings underscore the broader challenge of equipping ChatGPT with the skills to navigate the complexities of DSA problems effectively—a core aim of our investigation. Linguistic Abilities and Ethical Considerations

The inquiries by Zhuo et al. (2023) into ChatGPT's linguistic capabilities and ethical dimensions reveal concerns over bias, reliability, and the model's grasp of non-Latin scripts. These studies contribute to a critical discourse on the responsible development and deployment of LLMs, framing our research within a broader conversation about the ethical use of ChatGPT in educational and software development settings.

In sum, our literature review not only highlights the significant strides made in understanding and utilizing ChatGPT's capabilities but also identifies the gaps and challenges that persist, particularly in the realm of DSA. Our research seeks to bridge these gaps through a focused examination of prompt engineering as a tool for enhancing ChatGPT's problem-solving efficacy. By exploring the nuanced interaction between prompt specificity and model output, our study contributes to the ongoing dialogue on leveraging ChatGPT for more sophisticated and efficient software development processes.

2.1 Critique of the review

Our main findings from the literature review reveal that while significant strides have been made in understanding and applying ChatGPT in both coding and non-coding contexts, several key

areas require further exploration and development. Studies to date have effectively highlighted ChatGPT's capabilities and limitations, providing a solid foundation for assessing its applicability and performance in real-world tasks. However, a notable gap exists in the form of a lack of an automated pipeline to systematically test ChatGPT's efficacy on data structures and algorithms (DSA) coding questions.

Current research predominantly examines ChatGPT's performance through isolated demonstrations of its capabilities, lacking a comprehensive framework that would allow for ongoing evaluation and refinement across a range of DSA challenges. This gap points to the need for more structured research efforts that not only test but also refine ChatGPT's ability to handle complex coding problems. Moreover, the literature reveals that there is minimal focus on the development and testing of role-based prompts that could significantly enhance the model's utility by tailoring its responses to specific user roles or coding scenarios. This type of prompt engineering could increase the relevance and precision of ChatGPT's outputs, thereby improving its practical effectiveness in diverse programming environments.

This critique underscores the importance of advancing research to fill these gaps, particularly through the creation of an automated testing pipeline and the exploration of role-based prompt efficacy. Addressing these issues will be crucial for optimizing ChatGPT's performance and ensuring its readiness for more sophisticated and varied applications in the field of software development. Such advancements could dramatically enhance the model's functional utility, making it a more powerful tool in the arsenal of programmers and developers facing complex DSA challenges.

2.2 Summary

This chapter has underscored the significant progress in the deployment of Large Language Models like ChatGPT across diverse domains, with a particular focus on their application in software development and problem-solving within the realm of data structures and algorithms. The literature reviewed showcases both the strengths and limitations of ChatGPT, from basic CRUD operations to more complex coding tasks requiring nuanced understanding and creativity. Our analysis reveals that while ChatGPT exhibits promising capabilities, there are substantial areas for improvement, particularly in enhancing the model's precision and reliability through prompt engineering.

Furthermore, the absence of an automated testing pipeline for continuous evaluation and the potential for role-based prompts suggest promising directions for future research. By developing methods to systematically assess and refine ChatGPT's responses to DSA challenges, researchers can better harness its capabilities for efficient problem-solving. Ultimately, our study contributes to the ongoing discussion on improving and leveraging AI tools like ChatGPT, aiming to create more sophisticated and effective solutions in the field of software development.

Chapter 3

Methodology

This chapter details the comprehensive methodology employed in our research to assess the proficiency of ChatGPT 3.5 turbo, a state-of-the-art language model developed by OpenAI, in solving data structures and algorithms (DSA) coding problems using a variety of role-specific prompts. The process encompasses several phases including data collection, prompt engineering, solution processing, automated testing, and performance analysis, each designed to scrutinize the effectiveness of ChatGPT under different coding contexts.

3.1 Context and Objectives

The primary objective of this study is to create a streamlined and automated pipeline to explore how effectively ChatGPT can address DSA coding problems when guided by role-specific prompts that simulate different user roles or coding scenarios. This involves evaluating the model's ability to understand and generate appropriate code responses that successfully solve given problems. The study leverages a subset of problems from LeetCode, as mentioned in 1.4.1.

3.2 Data Collection

3.2.1 Fetching LeetCode Questions

Our data collection process deviates from traditional web scraping techniques. Instead, we utilized the LeetCode GraphQL API to programmatically fetch coding questions. This method provided a more stable and efficient means of extracting data, which includes question texts, metadata, code stubs, and public test cases. The dataset includes a total of 1,689 questions, categorized into 422 easy, 855 medium, and 412 hard questions.

3.2.2 Dataset Preparation

From the extensive pool of questions, we selected a balanced subset to ensure a comprehensive evaluation across various difficulty levels. The subset comprises 133 easy, 312 medium, and 124 hard questions. This selection was strategically made to challenge ChatGPT's coding capabilities under varying levels of complexity.

3.3 Prompt Engineering

3.3.1 Design and Implementation of Role-Specific Prompts

To simulate different coding scenarios and user roles, we designed nine distinct prompts. Each prompt was crafted to guide ChatGPT's response towards specific problem-solving strategies or viewpoints. These prompts are critical in evaluating how variations in query formulation can influence the effectiveness and accuracy of the solutions provided by ChatGPT.

3.3.2 Interaction with OpenAI's ChatGPT-3.5 Turbo

Each selected coding problem, accompanied by one of the nine role-specific prompts, was presented to ChatGPT-3.5 Turbo via the OpenAI API. This interaction was automated to handle the high volume of test cases, ensuring that each problem was processed efficiently and consistently.

3.4 Solution Processing and Testing

3.4.1 Automated Solution Adjustment

Upon receiving solutions from ChatGPT, a series of automated scripts were employed to refine the outputs. This refinement process included the removal of extraneous strings and docstrings, and the addition of necessary Python import statements to ensure that the code could be executed correctly in a Python environment.

3.4.2 Automated Testing Using Pytest

Following adjustment, solutions were automatically tested using Pytest, a powerful testing framework that verifies the functional correctness of code against the predefined test cases provided with the LeetCode problems. This testing phase is essential for objectively assessing the accuracy of the solutions generated by ChatGPT.

3.5 Performance Analysis

3.5.1 Data Aggregation

Results from the testing phase were systematically compiled into CSV files for each problem and corresponding prompt. These files contain detailed information including the number of tests passed, failed, and any errors encountered, providing a detailed breakdown of performance. The key metric observed in the study is acceptance rate which indicates whether all the test cases are passed without any errors.

3.5.2 Comprehensive Analysis

The aggregated performance data, along with the metadata initially scraped, were thoroughly analyzed to assess the impact of prompt specificity on ChatGPT's coding performance. This analysis focuses on identifying trends and deriving insights that could help optimize the use of AI for solving complex coding problems.

3.6 Summary

The methodology outlined in this chapter establishes a detailed and automated framework for evaluating the capabilities of ChatGPT in the realm of DSA problem-solving through targeted prompt engineering. By systematically exploring the interaction between the AI model and structured coding prompts, our study aims to contribute to the ongoing development of AI tools capable of effectively assisting in complex software development tasks.

Chapter 4

Results

This chapter presents the results obtained from the experimental evaluation of ChatGPT's performance in solving data structures and algorithms (DSA) problems using nine distinct role-based prompts. The analysis focuses on a subset of 569 leetcode problems categorized as 'Easy', 'Medium', and 'Hard', testing the effectiveness of each prompt in guiding ChatGPT to generate correct and efficient solutions.

4.1 Distribution of Problem Difficulty

Figure 4.1 shows the distribution of problems by difficulty level used in the experiments. This distribution was crucial in ensuring that the evaluation of each prompt's effectiveness was comprehensive, spanning simple to complex problem scenarios.

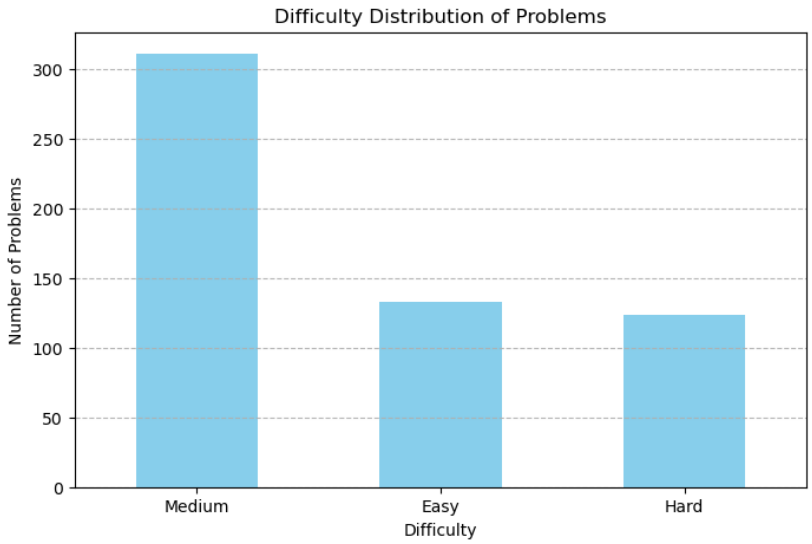

Figure 4.1: Distribution of problems by difficulty level

4.2 Overall Performance by Prompt

4.2.1 Accepted Solutions per Prompt

Table 4.1 provides the number of solutions accepted for each prompt. It is evident that certain prompts led to a higher number of accepted solutions, indicating their effectiveness in guiding ChatGPT to generate valid solutions.

Table 4.1: Number of Accepted Solutions per Prompt with Full Descriptions

Prompt ID	Prompt Description	Accepted Solutions
1	"Your role as an expert in Data Structures and Algorithms involves providing strictly accurate Python solutions. Provide only the code, with no extra text."	305
2	"You, a Python expert specializing in data structures and algorithms, are tasked with writing a function based on the stub, strictly following the problem's specifications for optimal time and space usage. Avoid any additional text, focusing solely on the code."	270
3	"You are an expert in Data Structures and Algorithms & you only give accurate solutions in python. You do not generate any additional text apart from the code."	300
4	"You are an expert Python developer focused on data structures and algorithms. Write an efficient Python function as provided in the code stub that adheres strictly to the problem's specifications and optimizes for both time and space. You do not generate any additional text or characters apart from the code."	340
5	"As a professional Python programmer specializing in data structures and algorithms, implement a solution for this LeetCode problem using the exact code signature provided. Ensure your code handles all constraints, edge cases and is optimized for performance. You do not generate any additional text apart from the code."	350
6	"You are tasked as a Python software engineer to develop code solving this algorithmic challenge. Write the cleanest and most efficient code, considering all given constraints and using the specified function names. You do not generate any additional text apart from the code."	360
7	"As a junior programmer, attempt to solve this problem in Python. Use what you've learned so far about data structures and algorithms. You do not generate any additional text apart from the code."	301
8	"You are a software engineering student. Please try to write Python code to solve this problem based on your current knowledge. You do not generate any additional text apart from the code."	346
9	"As an enthusiastic hobbyist coder, draft a Python script to tackle this algorithm problem efficiently. You do not generate any additional text apart from the code."	338

4.2.2 Graphical Representation of Results

Figure 4.2 provides a bar graph illustrating the total number of accepted solutions per prompt, which highlights the effectiveness of different prompt configurations in guiding the AI model.

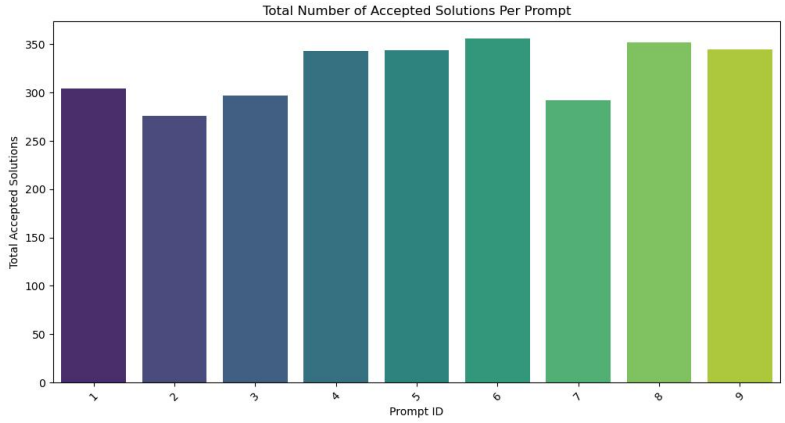

Figure 4.2: Total number of accepted solutions per prompt

4.3 Detailed Analysis of Prompt Performance

The performance of each prompt, particularly Prompt 6, is further detailed in Figure 4.13, where 62.7% of the test cases were accepted. This suggests a strong alignment between the prompt's specifications and ChatGPT's capability to deliver optimized solutions. Further analysis of the results by difficulty level is shown in Figure 4.14, which provides a clear breakdown of acceptance rates across easy, medium, and hard problem categories.

Generally, more number of errors are observed in the Hard category and a significantly lower number observed in the Easy category as shown in the consequent figures. This can be attributed to not only the difficulty level of problems but also, more specifically it points out that ChatGPT 3.5 turbo is in-adept in tackling the "twists" (twists in leetcode DSA questions refers to the re-framing of the problem in it's complicated form) observed in the more difficult problems.

Upon taking a closer look at the best performing prompt i.e., prompt 6, it reveals that specificity of what is required from the model in the context of DSA questions in order to solve them effectively plays a pivotal role, which lacks to certain degree in some other promps. In essence, vagueness and abstract wording do not help.

However, results also show (see fig. 5.1) that prompting increases the models solving capabilities significantly and for some topics, even doubling the general acceptance rate. This shows the effectiveness of prompting the right contexts in solving DSA questions.

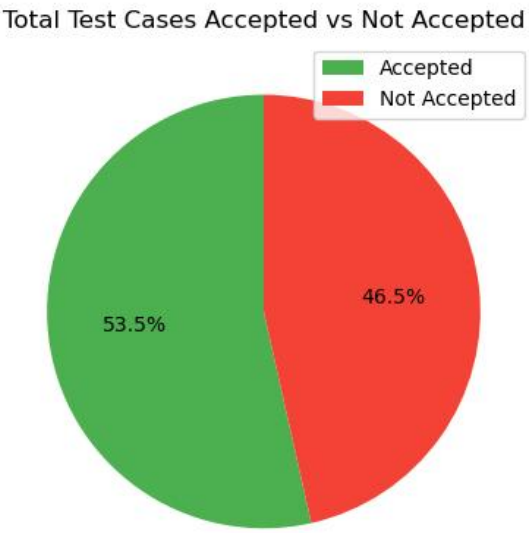

Figure 4.3: Performance breakdown for Prompt 1

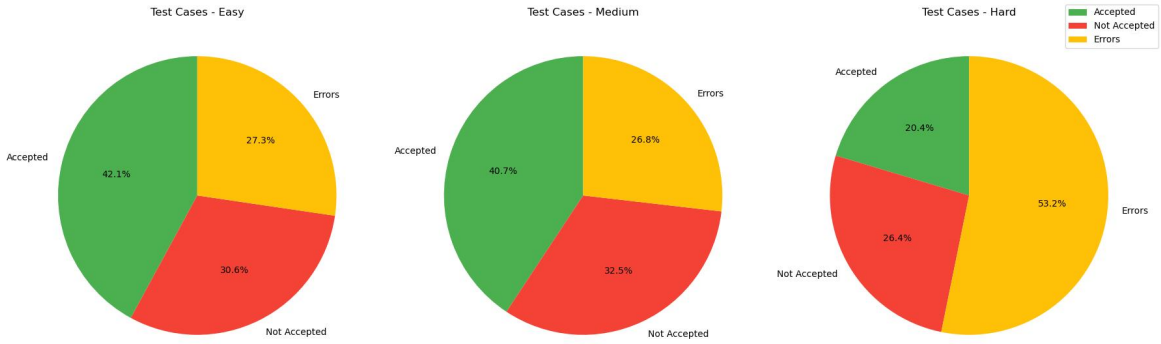

Figure 4.4: Distribution of test case results by difficulty level for Prompt 1

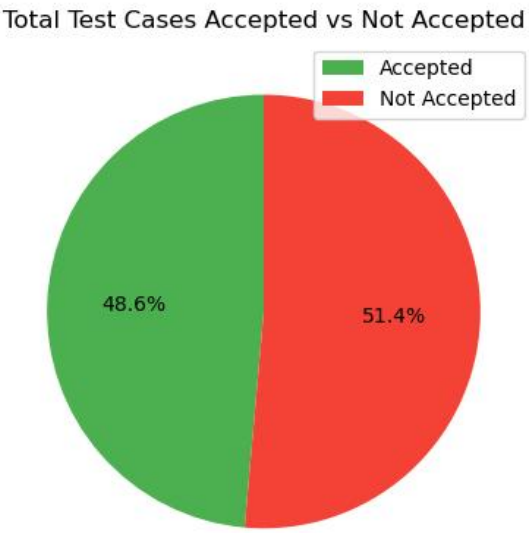

Figure 4.5: Performance breakdown for Prompt 2

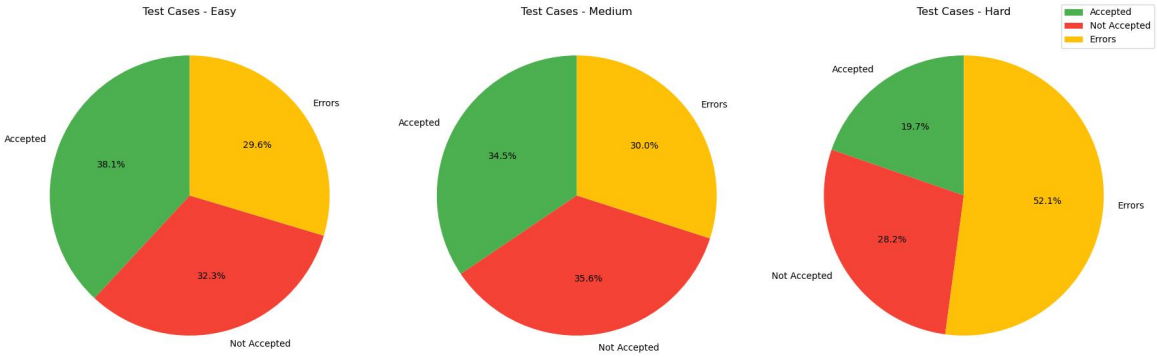

Figure 4.6: Distribution of test case results by difficulty level for Prompt 2

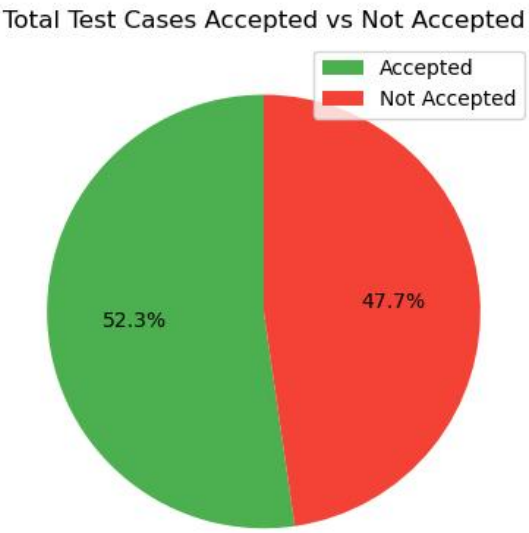

Figure 4.7: Performance breakdown for Prompt 3

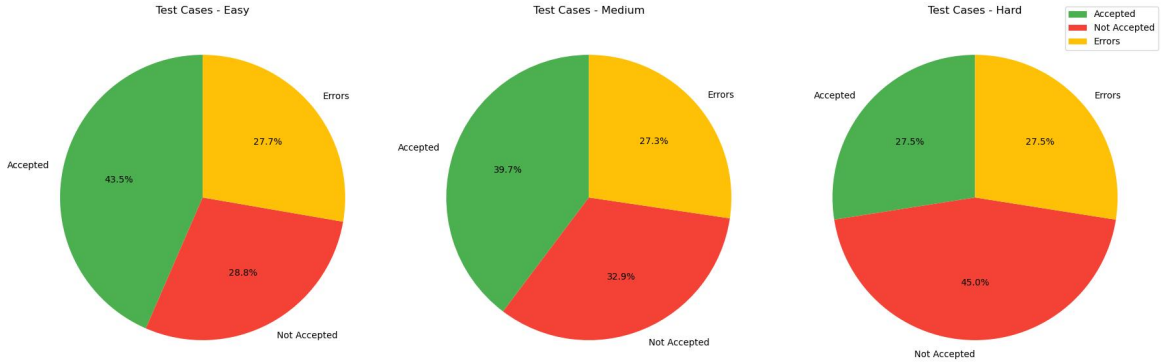

Figure 4.8: Distribution of test case results by difficulty level for Prompt 3

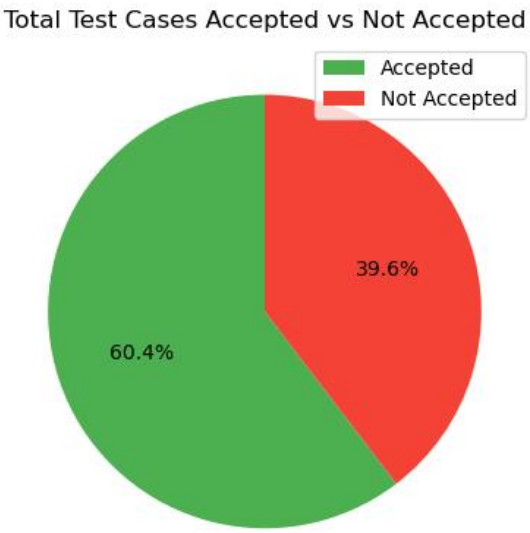

Figure 4.9: Performance breakdown for Prompt 4

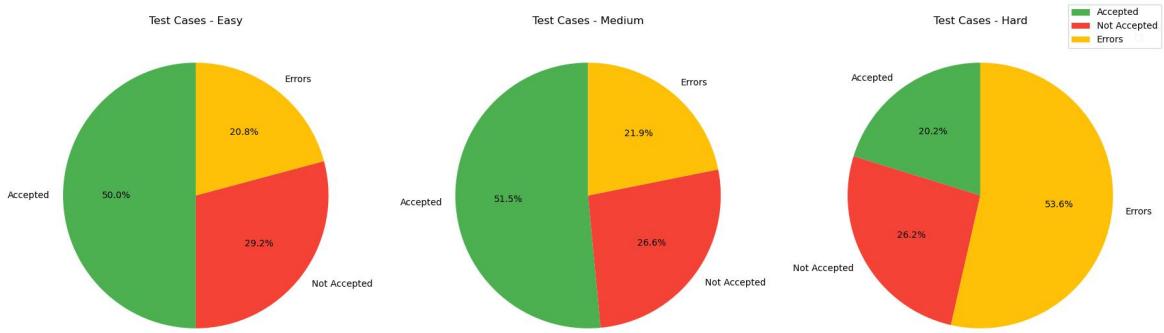

Figure 4.10: Distribution of test case results by difficulty level for Prompt 4

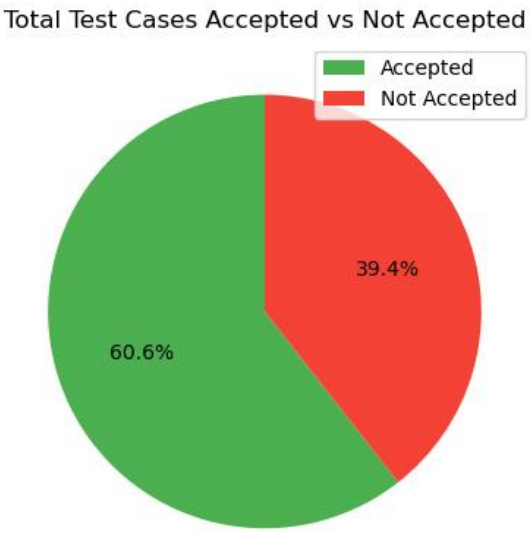

Figure 4.11: Performance breakdown for Prompt 5

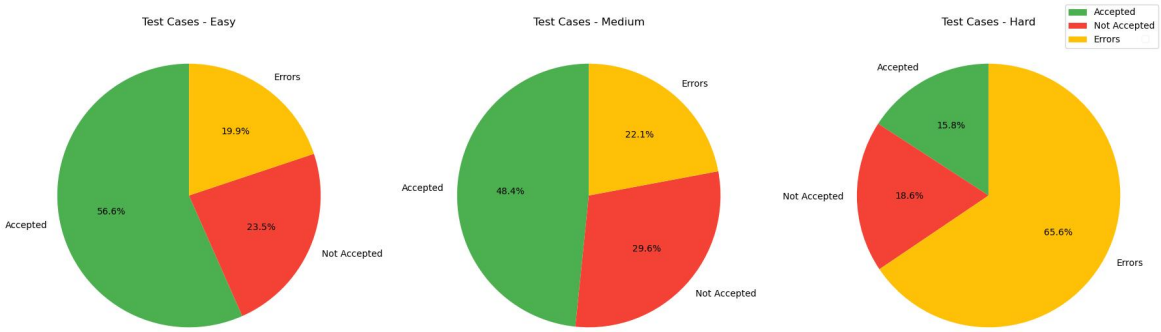

Figure 4.12: Distribution of test case results by difficulty level for Prompt 5

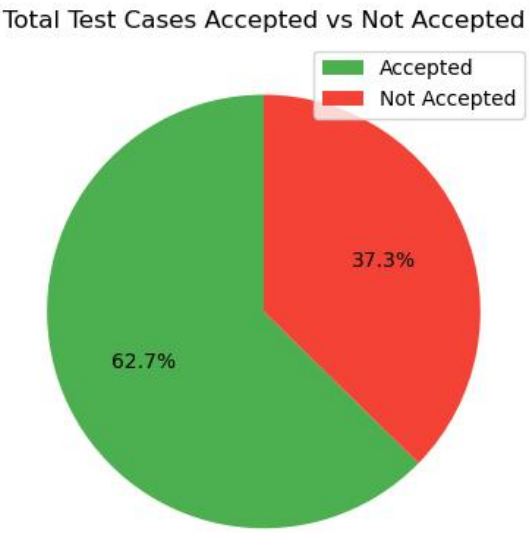

Figure 4.13: Performance breakdown for Prompt 6

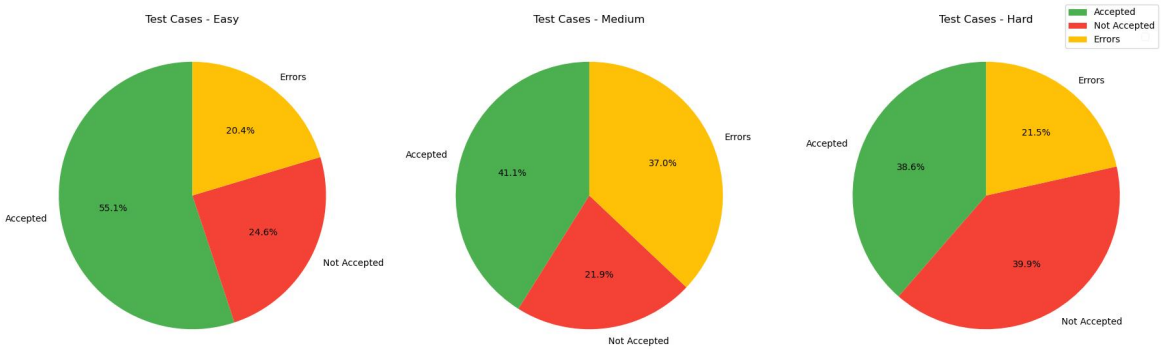

Figure 4.14: Distribution of test case results by difficulty level for Prompt 6

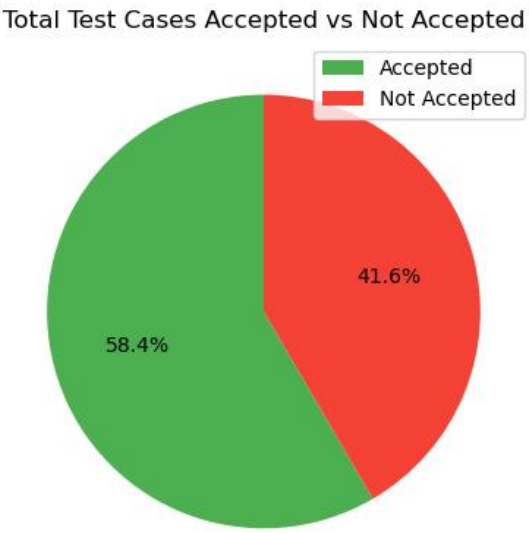

Figure 4.15: Performance breakdown for Prompt 7

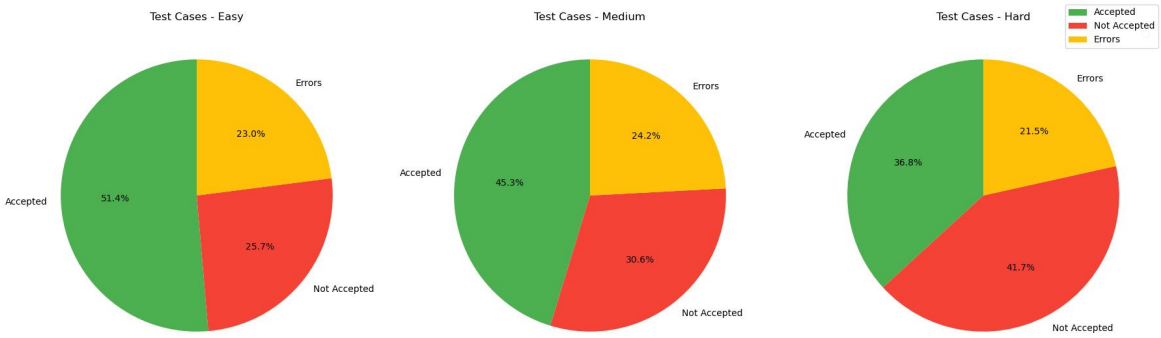

Figure 4.16: Distribution of test case results by difficulty level for Prompt 7

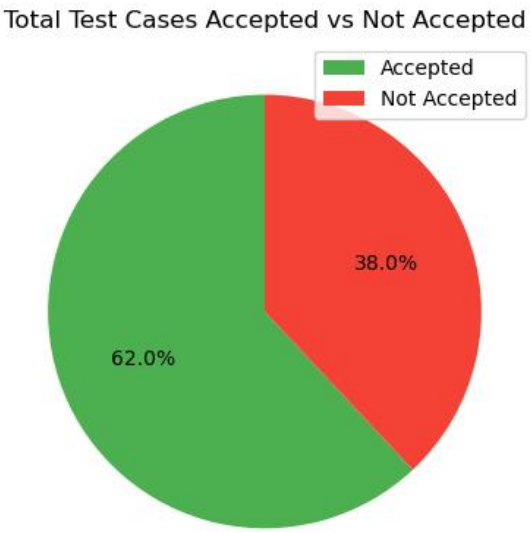

Figure 4.17: Performance breakdown for Prompt 8

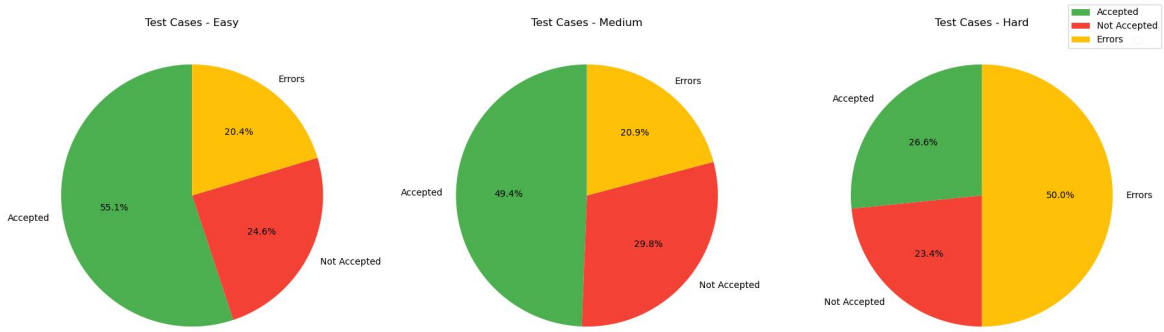

Figure 4.18: Distribution of test case results by difficulty level for Prompt 8

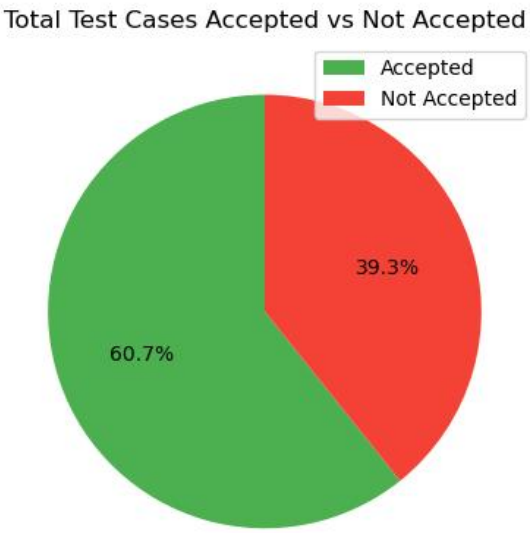

Figure 4.19: Performance breakdown for Prompt 9

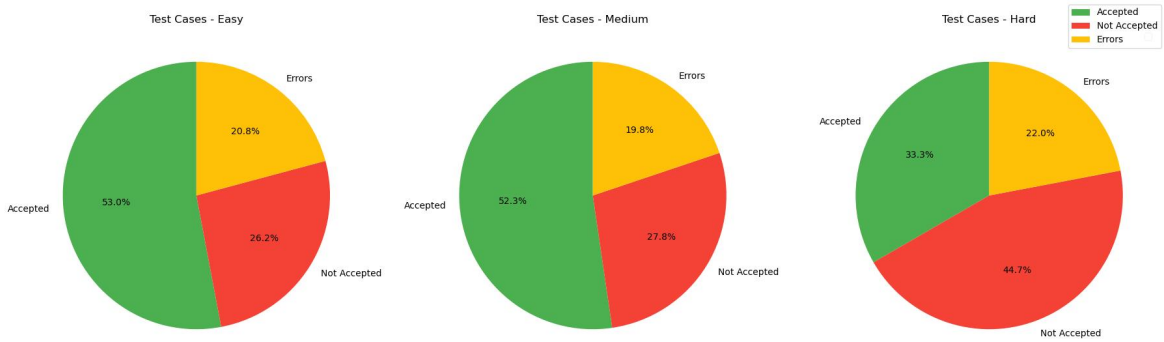

Figure 4.20: Distribution of test case results by difficulty level for Prompt 9

4.3.1 Observing erroneous patterns

The following figures 4.4, 4.6, 4.8, 4.10, 4.12, 4.14, 4.16, 4.18, 4.20 reveal the erroneous code generation which leads to failure. A simple correlation can be observed between the total number of errors to the overall acceptance rate. This suggests that guiding ChatGPT to produce less errors can be the key to unlocking higher orders of performance. Further study and analysis needs to be done to identify the causes for errors and the effect of prompting in guiding the model to produce error free code.

4.4 Summary

The results presented in this chapter reveal significant insights into the effectiveness of different prompts in guiding ChatGPT to successfully solve DSA problems. The analysis indicates that specific prompts, especially those requiring concise and precise code solutions, tend to yield higher acceptance rates. This chapter sets the stage for a deeper discussion on the implications of these findings, which will be explored in the subsequent chapters of the thesis. For a detailed analysis of acceptance rates per topic for each prompt, refer to Appendix A.

Chapter 5

Discussion and Analysis

This chapter delves into the insights drawn from the results presented in the previous chapter, evaluating the performance of ChatGPT in generating solutions for data structures and algorithms (DSA) problems across various topics and under different prompts. The discussion aims to interpret the significance of the findings, highlight key limitations, and suggest avenues for future research.

5.1 Impact of Keywords in Prompts on Acceptance Rates

The effectiveness of the prompts in guiding ChatGPT towards successful solutions is notably influenced by specific keywords. For instance, the inclusion of the word "efficient" in Prompts 4, 6, and 9 correlated with a higher number of accepted solutions, as shown in Table 4.1. This suggests that directing ChatGPT to focus on efficiency not only in code execution but also in resource utilization resonates well with the model's capabilities in optimizing solutions.

5.2 Role Influence on Prompt Performance

Analysis of the role-specified in each prompt reveals a significant impact on performance. Prompts that described the user as an "expert" or "software engineer" tended to perform better, implying that a higher expectation in the role description possibly aligns better with ChatGPT's training on professional and technical language. Conversely, prompts that positioned the user as a "junior programmer" or "student" saw slightly lower acceptance rates, suggesting a mismatch between the complexity of problems and the assumed knowledge base of the role.

5.3 Topic-wise Acceptance Rates

Figure 5.1 presents a comparison of provided versus calculated acceptance rates across various DSA topics. The provided rates represent the expected success rates based on historical data, while the calculated rates are derived from ChatGPT's performance across nine prompts.

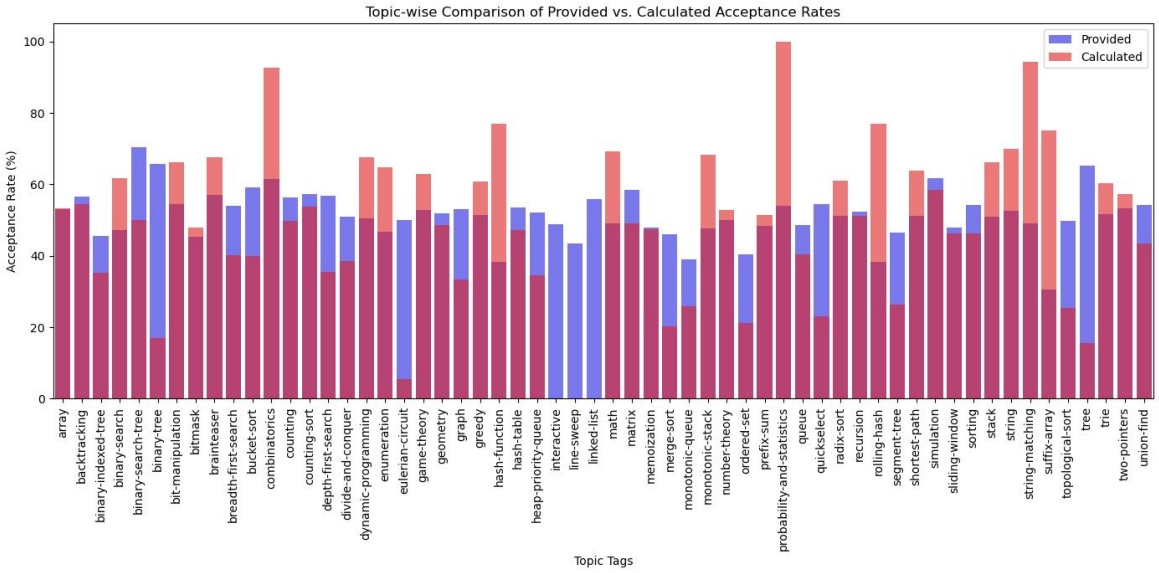

Figure 5.1: Topic-wise comparison of provided vs. calculated acceptance rates

ChatGPT exhibited exceptional performance in topics related to probability, statistics, combinatorics, hash function, rolling hash, string matching, and suffix array. These topics often involve a significant amount of mathematical reasoning and pattern recognition, areas where machine learning models like ChatGPT are particularly strong. On the other hand, the model struggled with topics such as binary tree, eulerian circuit, line sweep, linked list, and tree, which require a deep understanding of data structure manipulation and spatial relationships—a current limitation of language-based AI models.

5.4 Implications of Findings

The findings underscore the importance of prompt design in the effective utilization of AI for problem-solving in coding. By aligning the prompts with ChatGPT's training and capabilities, and by specifically targeting its strengths in mathematical reasoning and pattern recognition, the efficiency of AI-assisted coding solutions can be significantly enhanced.

5.5 Limitations and Future Work

While the study provides valuable insights into the capabilities of ChatGPT, it also highlights limitations, particularly in handling complex data structures. Future work could explore more granular prompt engineering, incorporate visual or spatial data representation techniques, and extend the analysis to newer versions of language models that might offer improved understanding of complex algorithms and data structures.

5.6 Summary

This chapter has discussed the key findings from the evaluation of ChatGPT's performance in solving DSA problems, highlighting how specific keywords and role expectations in prompts influence the effectiveness of the solutions generated. The analysis of topic-wise acceptance rates has revealed areas where ChatGPT excels and struggles, providing a roadmap for both leveraging its strengths and addressing its weaknesses in future applications.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

This thesis has systematically investigated the performance of ChatGPT 3.5 Turbo, an advanced language model developed by OpenAI, in addressing complex data structures and algorithms (DSA) problems. Over the course of this study, a diverse array of problems categorized by varying levels of difficulty was approached using role-based prompts, specifically designed to probe the model's ability to adapt and respond to nuanced programming tasks.

The experimentation was rooted in a set of role-based prompts, each tailored with specific keywords and structured to simulate different programming expertise levels—from a junior programmer to an expert in Python programming. The analysis revealed a significant correlation between the effectiveness of these prompts and the performance outcomes, with prompts that emphasized "efficiency" and advanced programming expertise consistently yielding higher acceptance rates. Such findings demonstrate the profound impact that targeted prompt engineering can have on the operational effectiveness of language models like ChatGPT.

Interestingly, the study uncovered that while ChatGPT demonstrates exceptional strength in solving problems that require strong mathematical reasoning and advanced pattern recognition, it faces notable challenges when tasked with problems that demand a deep understanding of complex data structures and the manipulation of spatial relationships. This suggests a potential limitation in its current training, which seems less equipped to handle tasks that go beyond text-based reasoning to require a conceptual grasp of spatial and structural dynamics.

Moreover, this research has not only highlighted the specific strengths and weaknesses of ChatGPT in the realm of programming and problem-solving but has also set a precedent for the use of automated pipelines in evaluating AI models. The automated pipeline developed for this study enables seamless, efficient testing of a large number of problems across various difficulty settings without manual intervention. This system is instrumental in scaling the analysis to accommodate a wider range of prompts and scenarios, making it a valuable tool for future research.

The flexibility and scalability provided by the automated pipeline pave the way for conducting similar analyses on different large language models (LLMs) without the need for extensive manual setup. This capability is crucial for comparative studies that aim to benchmark the performance of various AI models under identical conditions, thereby providing a clearer picture of their relative strengths and limitations in technical domains.

In summary, the work presented in this thesis underscores the nuanced interplay between prompt design and AI performance, emphasizing the importance of precise and thoughtful prompt engineering in extracting the maximum potential from AI capabilities. The findings from this study not only contribute to the ongoing discourse on the application of AI in coding and programming challenges but also provide a framework for future explorations aimed at enhancing the robustness and versatility of AI solutions in software development and beyond.

6.2 Future Work

Reflecting on the findings and limitations of this study, several areas for future research have been identified to extend and enhance the utility of language models like ChatGPT in solving complex problems:

1. **Expanded Prompt Testing:** Further testing with a broader array of prompts, especially those that might explicitly guide the model through complex data structure manipulations, could provide deeper insights into enhancing model performance. This could involve exploring more nuanced and varied prompt structures to better engage the model's capabilities.

2. **Increased Data Diversity:** Utilizing a larger and more varied dataset could help in generalizing the findings of this study and exploring the scalability of the proposed solutions. This would involve incorporating a wider range of problem types and complexities to robustly test the model's adaptability and accuracy.

3. **Alternative Language Models:** Experimenting with different language models or newer versions of ChatGPT could reveal improvements or variations in performance across different AI architectures. This includes testing models that have been trained with different datasets or optimized for specific types of tasks.

4. **Integration of Visual or Spatial Data Representation Techniques:** For problems involving spatial relationships, integrating visual aids or spatial data representations might help overcome some of the current limitations. This could involve using diagrams or other visual inputs to aid the model in understanding and solving spatially-related problems.

5. **Granular Analysis of Error Types:** Identifying specific categories of errors and their causes could lead to more targeted improvements in model training and prompt design. This analysis would help in refining the prompts or training methods to minimize errors and enhance the model's problem-solving accuracy.

6. **Limitations Due to API Constraints:** The OpenAI API currently limits the number of questions to 540 per prompt, which constrains the volume of data that can be processed in a single batch. This limitation could impact the comprehensiveness of testing and benchmarking efforts. Future work could explore ways to manage or circumvent these limitations, perhaps by varying prompts or partitioning datasets to allow more exhaustive testing.

These directions not only aim to address the gaps identified through this research but also to leverage emerging technologies and methodologies to further the capabilities of AI in programming and problem-solving contexts.

Chapter 7

Reflection

This project has been a profound journey into the capabilities and limitations of artificial intelligence, specifically within the context of solving data structures and algorithms (DSA) problems using ChatGPT. Reflecting on this experience, I recognize it as not only a technical exploration but also as a significant period of personal and professional growth.

One of the initial challenges I faced was the lack of libraries or open-source resources specifically tailored for gathering coding problems and their test cases. This limitation necessitated a deep dive into existing methodologies, which often fell short in directly addressing the needs of this project. The absence of a ready-made solution for fetching diverse and complex coding questions led me to develop a new approach for data acquisition. This process involved exploring various databases and APIs, eventually leading to the creation of an automated pipeline that utilized the LeetCode GraphQL API. This experience was invaluable as it pushed me to innovate and create tools that were not only useful for this project but also adaptable for future research in this field.

Throughout this project, I learned the critical importance of role-based prompt engineering. By crafting prompts that simulated different user roles—from a junior programmer to an expert in Python programming—I was able to significantly influence ChatGPT's output. This exploration revealed how subtle changes in language and the inclusion of specific keywords could dramatically alter the effectiveness of the solutions provided by the AI. The experience has sharpened my skills in understanding how to guide AI to perform tasks that it might not inherently excel at, especially in areas requiring a deep understanding of complex data structures.

A substantial portion of my time was devoted to analyzing the responses generated by ChatGPT, identifying patterns, and fine-tuning the post-processing of these solutions to ensure they could be executed on test cases without human intervention. This aspect of the project was particularly challenging due to the variability in the AI's responses. Developing a methodical approach to automatically adjust and correct the generated code was both challenging and rewarding. It underscored the importance of iterative testing and adjustment in AI-related projects, where outcomes can be unpredictable and require continuous refinement.

Looking back, if I were to tackle a similar problem in the future, I would place a greater emphasis on developing even more robust tools for data handling and analysis from the outset. The iterative nature of AI testing and the variability in model performance highlighted the need for flexible and adaptable project planning.

Moreover, this project has significantly impacted my understanding of the role of AI in educational and professional settings. It has provided me with a solid foundation in research method-

ology, enhanced my problem-solving skills, and deepened my appreciation for the detailed and careful design required to successfully implement AI solutions. The insights gained from this project will undoubtedly influence my approach to future AI research and applications, particularly in optimizing the interaction between human input and AI output.

Overall, this reflective summary encapsulates not just a list of technical skills acquired but also the broader learning experiences—highlighting challenges, adaptations, and personal insights gained throughout the duration of the project.

References

Arefin, S. E., Heya, T. A., Al-Qudah, H., Ineza, Y. and Serwadda, A. (2023), 'Unmasking the giant: A comprehensive evaluation of chatgpt's proficiency in coding algorithms and data structures'.

Biswas, S. (2023), 'Role of chatgpt in computer programming'.

Bubeck, S., Chandrasekaran, V., Eldan, R., Gehrke, J., Horvitz, E., Kamar, E., Lee, P., Lee, Y. T., Li, Y., Lundberg, S., Nori, H., Palangi, H., Ribeiro, M. T. and Zhang, Y. (2023), 'Sparks of artificial general intelligence: Early experiments with gpt-4'.

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. d. O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G. et al. (2021), 'Evaluating large language models trained on code', *arXiv preprint arXiv:2107.03374* .

Feng, Y., Vanam, S., Cherukupally, M., Zheng, W., Qiu, M. and Chen, H. (2023), Investigating code generation performance of chatgpt with crowdsourcing social data, *in* '2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC)', pp. 876–885.

Frieder, S., Pinchetti, L., Chevalier, A., Griffiths, R.-R., Salvatori, T., Lukasiewicz, T., Petersen, P. C. and Berner, J. (2023), 'Mathematical capabilities of chatgpt'.

Gilson, A., Safranek, C. W., Huang, T., Socrates, V., Chi, L., Taylor, R. A. and Chartash, D. (2023), 'How does chatgpt perform on the united states medical licensing examination? the implications of large language models for medical education and knowledge assessment', *JMIR Med Educ* **9**, e45312.
URL: *https://mededu.jmir.org/2023/1/e45312*

Noever, D. and McKee, F. (2023), 'Numeracy from literacy: Data science as an emergent skill from large language models'.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I. et al. (2019), 'Language models are unsupervised multitask learners', *OpenAI blog* **1**(8), 9.

Tian, H., Lu, W., Li, T. O., Tang, X., Cheung, S.-C., Klein, J. and Bissyandé, T. F. (2023), 'Is chatgpt the ultimate programming assistant – how far is it?'.

Zhuo, T. Y., Huang, Y., Chen, C. and Xing, Z. (2023), 'Red teaming chatgpt via jailbreaking: Bias, robustness, reliability and toxicity'.

Appendix A

An Appendix Chapter

A.1 Topic-Wise Acceptance Rates for Each Prompt

This section presents the topic-wise acceptance rates for each of the nine prompts used in the study. These figures illustrate how the model performed across various computer science topics, highlighting areas of strength and weakness for each prompt. These visualizations serve as a valuable tool for understanding the specific strengths and weaknesses of the AI model in solving diverse coding problems, and it can guide future research and prompt engineering efforts to enhance model performance.

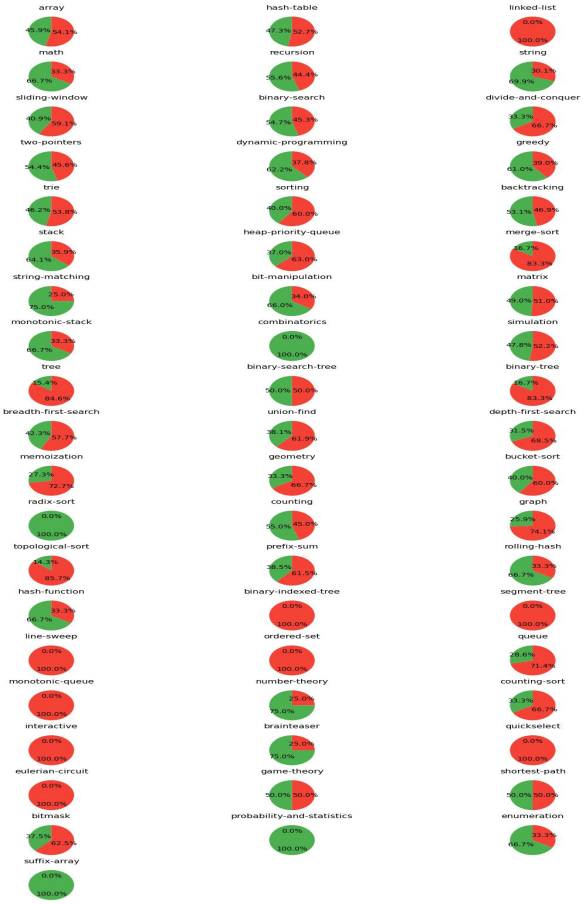

Figure A.1: Topic-wise acceptance rate for Prompt 1, illustrating the percentage of solutions accepted (green) and not accepted (red) across different undergraduate computer science topics.

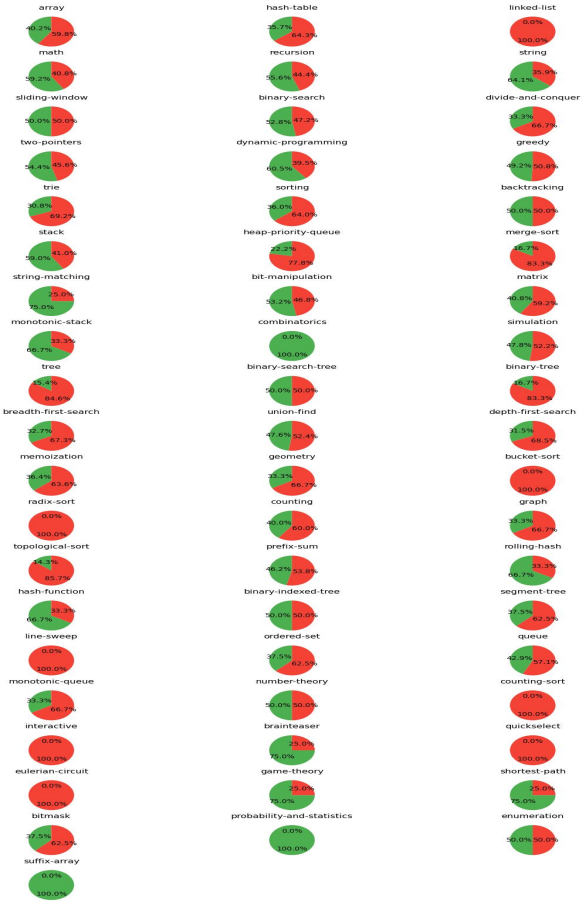

Figure A.2: Topic-wise acceptance rate for Prompt 2, illustrating the percentage of solutions accepted (green) and not accepted (red) across different undergraduate computer science topics.

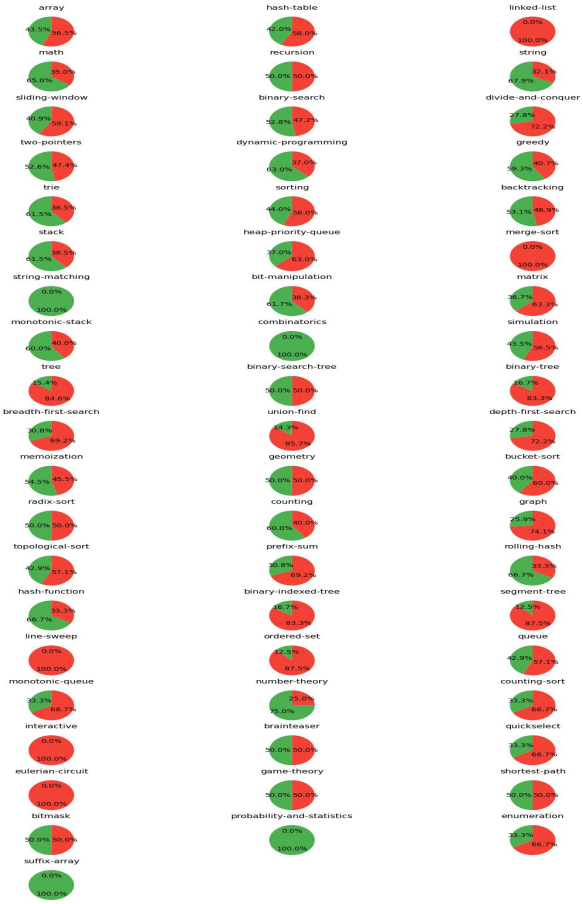

Figure A.3: Topic-wise acceptance rate for Prompt 3, illustrating the percentage of solutions accepted (green) and not accepted (red) across different undergraduate computer science topics.

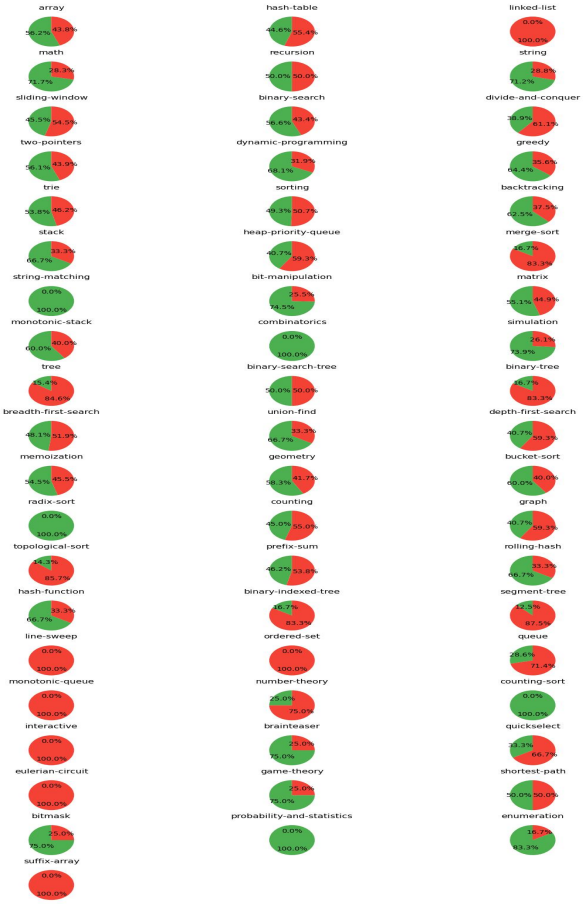

Figure A.4: Topic-wise acceptance rate for Prompt 4, illustrating the percentage of solutions accepted (green) and not accepted (red) across different undergraduate computer science topics.

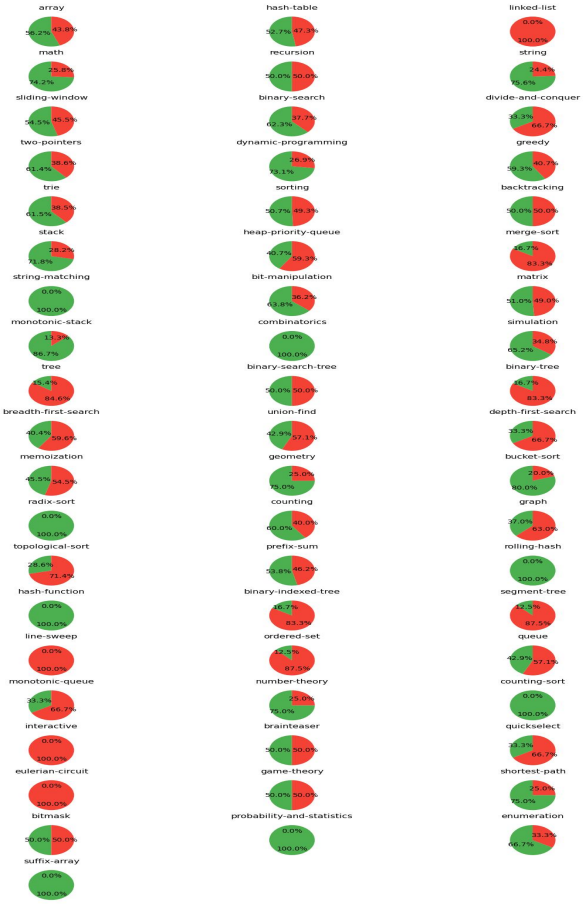

Figure A.5: Topic-wise acceptance rate for Prompt 5, illustrating the percentage of solutions accepted (green) and not accepted (red) across different undergraduate computer science topics.

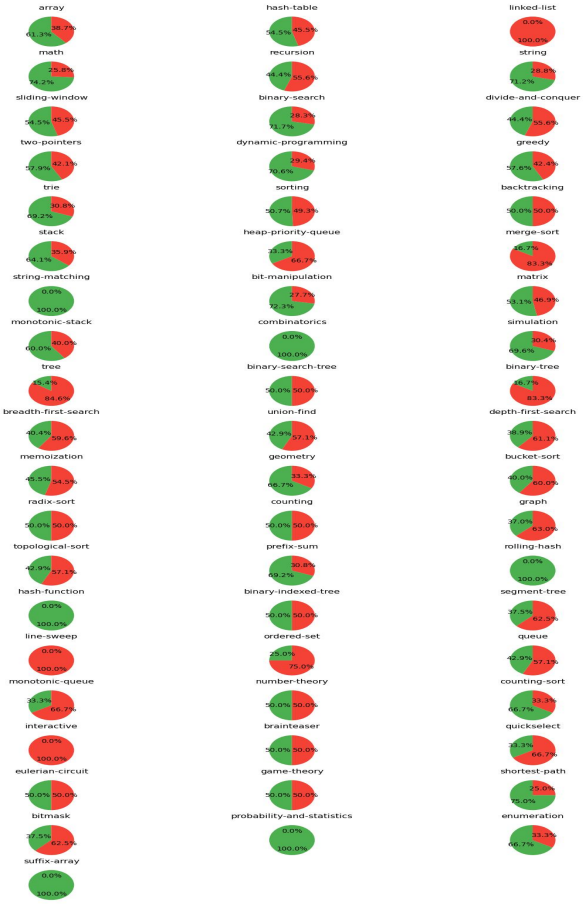

Figure A.6: Topic-wise acceptance rate for Prompt 6, illustrating the percentage of solutions accepted (green) and not accepted (red) across different undergraduate computer science topics.

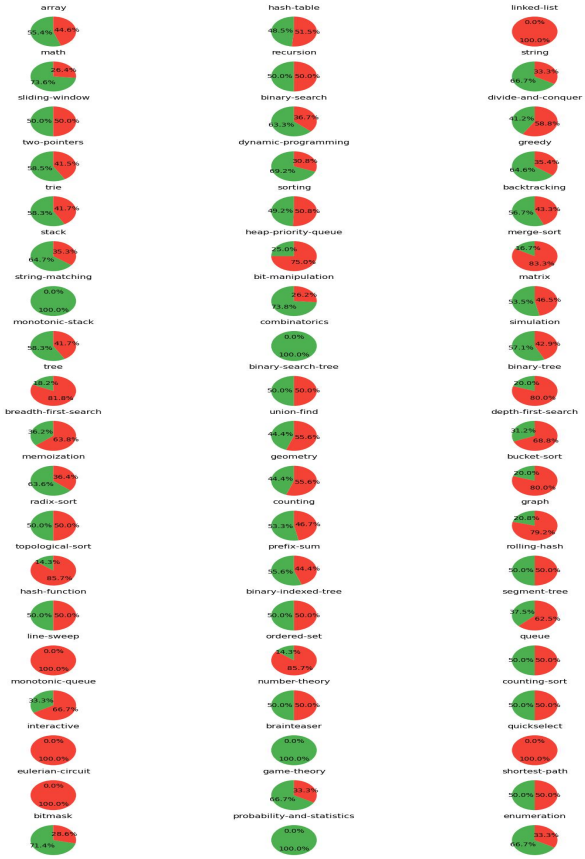

Figure A.7: Topic-wise acceptance rate for Prompt 7, illustrating the percentage of solutions accepted (green) and not accepted (red) across different undergraduate computer science topics.

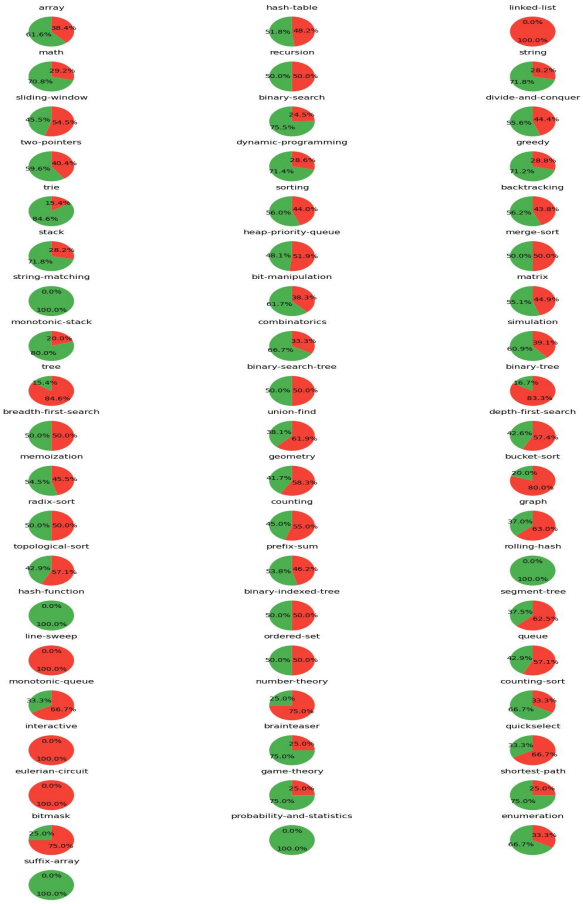

Figure A.8: Topic-wise acceptance rate for Prompt 8, illustrating the percentage of solutions accepted (green) and not accepted (red) across different undergraduate computer science topics.

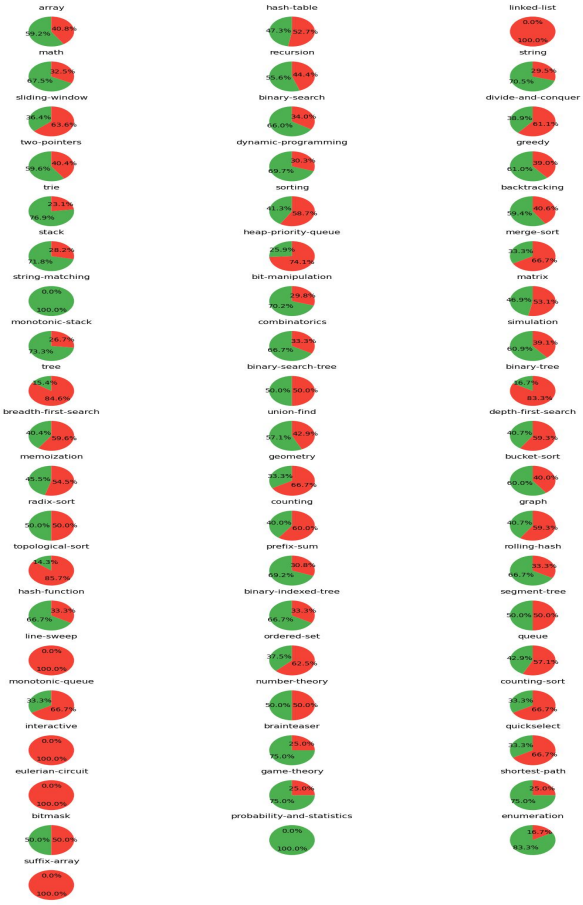

Figure A.9: Topic-wise acceptance rate for Prompt 9, illustrating the percentage of solutions accepted (green) and not accepted (red) across different undergraduate computer science topics.