IJIRSET
International Journal of Innovative Research in
SCIENCE | ENGINEERING | TECHNOLOGY

e-ISSN: 2319-8753 | p-ISSN: 2347-6710

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

Volume 13, Issue 11, November 2024

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.524

9940 572 462 | 6381 907 438 | ijirset@gmail.com | www.ijirset.com

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.524

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 13, Issue 11, November 2024

|DOI: 10.15680/IJIRSET.2024.1311087|

Lean Module for Reasoning about Computation Complexity in GPTs

P. Sahithi[1], Dileep Kumar[2], Abinav Ram[3], Mr. D A Kiran Kumar[4]

Department of Computer Science Engineering, Anurag University, Hyderabad, Telangana, Hyderabad, India[1,2,3]

Assistant Professor, Department of Computer Science and Engineering, Anurag University, Hyderabad, Telangana, India[4]

ABSTRACT: Analyzing and estimating algorithm complexity is a cornerstone of computational science, essential for evaluating the efficiency and scalability of algorithms in diverse applications. This study presents an interactive module for classifying algorithm complexity using common terms and identifying corresponding Big O notations. Implemented as a web-based tool using Streamlit, the module allows users to input a problem statement and algorithm description to automatically determine the complexity class and associated time complexity notation (Big O). It categorizes algorithms into classes such as exponential, polynomial, logarithmic, and linear by analyzing key keywords and structural patterns in the description.

The tool enhances accessibility for students, researchers, and practitioners by simplifying complexity analysis, offering instant feedback, and suggesting refinements for improving algorithm performance based on time and space complexity assessments. Although the module currently relies on keyword-based analysis, this paper outlines future enhancements incorporating advanced computational complexity models and automated pattern recognition using machine learning to improve classification accuracy. The simplicity and adaptability of this tool demonstrate its potential as an educational aid in computer science curricula and as a productivity resource for software developers.

KEYWORDS: Analysing complexities, Big O notaion, User-friendly Platform, Exponential,Logarithmic,Polynomial.

I. INTRODUCTION

The efficiency of algorithms plays a critical role in software development and data processing, particularly as systems scale in size and complexity. Understanding the time and space complexity of algorithms enables developers and researchers to make informed decisions regarding performance optimization, resource allocation, and scalability. Complexity analysis also aids in evaluating potential trade-offs in algorithm design, which are essential for applications in machine learning, data science, and real-time processing. However, many students and novice developers face challenges in manually determining the complexity class of an algorithm, particularly when analyzing unfamiliar code structures or theoretical concepts.

To address this gap, this research introduces a user-friendly, web-based module for analyzing algorithm complexity, developed with Streamlit, a popular Python library for creating interactive applications. This module allows users to input descriptions of algorithms and, based on key characteristics, automatically classifies the algorithms into commonly known complexity categories: exponential, polynomial, logarithmic, and linear. By offering insights into the time and space complexity—expressed in Big O notation—the tool provides users with a practical means of understanding algorithmic efficiency. It classifies complexities based on keywords such as "recursive," "loop," and "search," thus offering a simplified approach to categorizing algorithms that does not require advanced mathematical calculations.

This research explores the design and development of the module, demonstrating its utility as an educational aid and productivity tool. By reducing the cognitive load associated with complexity analysis, the module aims to empower students, educators, and early-career developers to assess algorithm performance effectively. The current model is rule-based and relies on keyword detection, but this paper also outlines the potential for future iterations that incorporate machine learning algorithms to improve classification accuracy and handle more complex scenarios.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.524

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 13, Issue 11, November 2024

|DOI: 10.15680/IJIRSET.2024.1311087|

II. RESEARCH METHODOLOGY

The methodology for developing this algorithm complexity analysis tool is based on a structured approach that includes the design, implementation, validation, and evaluation phases. Each stage is carefully outlined below to illustrate the steps taken in building, testing, and refining the tool.

1. Design and Framework Selection
The initial phase of the research involved identifying a suitable platform for building an interactive web-based tool. Streamlit was chosen for its simplicity, efficiency, and popularity in the Python community, especially for creating data science applications. The framework's flexibility allows for rapid prototyping and deployment, enabling an efficient process for receiving user feedback and improving the tool iteratively.

2. Algorithm Complexity Classification Model
The tool's core functionality relies on a rule-based classification model that interprets the algorithm's description text and assigns a complexity category. A keyword-based approach was adopted to classify algorithms into broad complexity classes (e.g., exponential, polynomial, logarithmic, and linear) based on typical patterns observed in algorithm structures:

- **Recursive Structures**: Recognized as having exponential complexity (e.g., O(2^n)).
- **Loop Structures**: Classified as polynomial complexity, often quadratic or cubic (e.g., O(n^2)).
- **Search Operations**: Mapped to logarithmic complexity (e.g., O(log n)), based on common binary search patterns.
- **Simple Linear Structures**: Identified as having linear complexity (e.g., O(n)), typical of algorithms that iterate once over input data.

This approach provides a baseline for algorithm classification without requiring advanced mathematical analysis, making it accessible for users with varying levels of computational expertise.

3. Conversion to Big O Notation
To make the complexity more interpretable, each classified category is mapped to a Big O notation representation, a standard mathematical convention for describing asymptotic complexity. The classification model translates recognized structures into their Big O notation, helping users understand algorithmic efficiency in a commonly accepted format.

4. Development of the Computational Evaluation Process
In addition to classifying time complexity, the tool provides a preliminary evaluation of space complexity. For simplicity, the current model assumes a linear space complexity (O(n)) as a placeholder; however, future enhancements may involve more nuanced space complexity analysis. The tool presents this information interactively, allowing users to enter an algorithm description and receive a breakdown of both time and space complexities.

5. Validation and Testing
Validation involved testing the tool against a set of common algorithms with known complexities to ensure accuracy. Algorithms were selected to represent each of the primary complexity classes the tool aims to identify, including recursive functions, iterative loops, search algorithms, and linear operations. By comparing the tool's classifications against established theoretical complexities, we assessed the accuracy and reliability of the rule-based model.

III. THEORY

The algorithm developed in this project aims to automate the complexity analysis of algorithms by leveraging natural language processing and rule-based classification. Using the Streamlit library, we built a web-based application that takes user input in the form of algorithm descriptions, parses this information, and assigns a complexity class based on recognized structural patterns.

The tool primarily focuses on identifying common keywords and structures to infer algorithm complexity. This includes detecting recursive patterns indicative of exponential complexity, iterative loops suggesting polynomial complexity, search terms associated with logarithmic complexity, and simple linear operations that point to linear

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.524

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 13, Issue 11, November 2024

|DOI: 10.15680/IJIRSET.2024.1311087|

complexity. By analyzing these structural elements, the tool classifies algorithms into complexity classes that are then mapped to Big O notation, providing a theoretical measure of time complexity.

Exponential Complexity (O(2^n)): Commonly found in recursive algorithms, this complexity reflects algorithms where each additional input exponentially increases the number of operations.

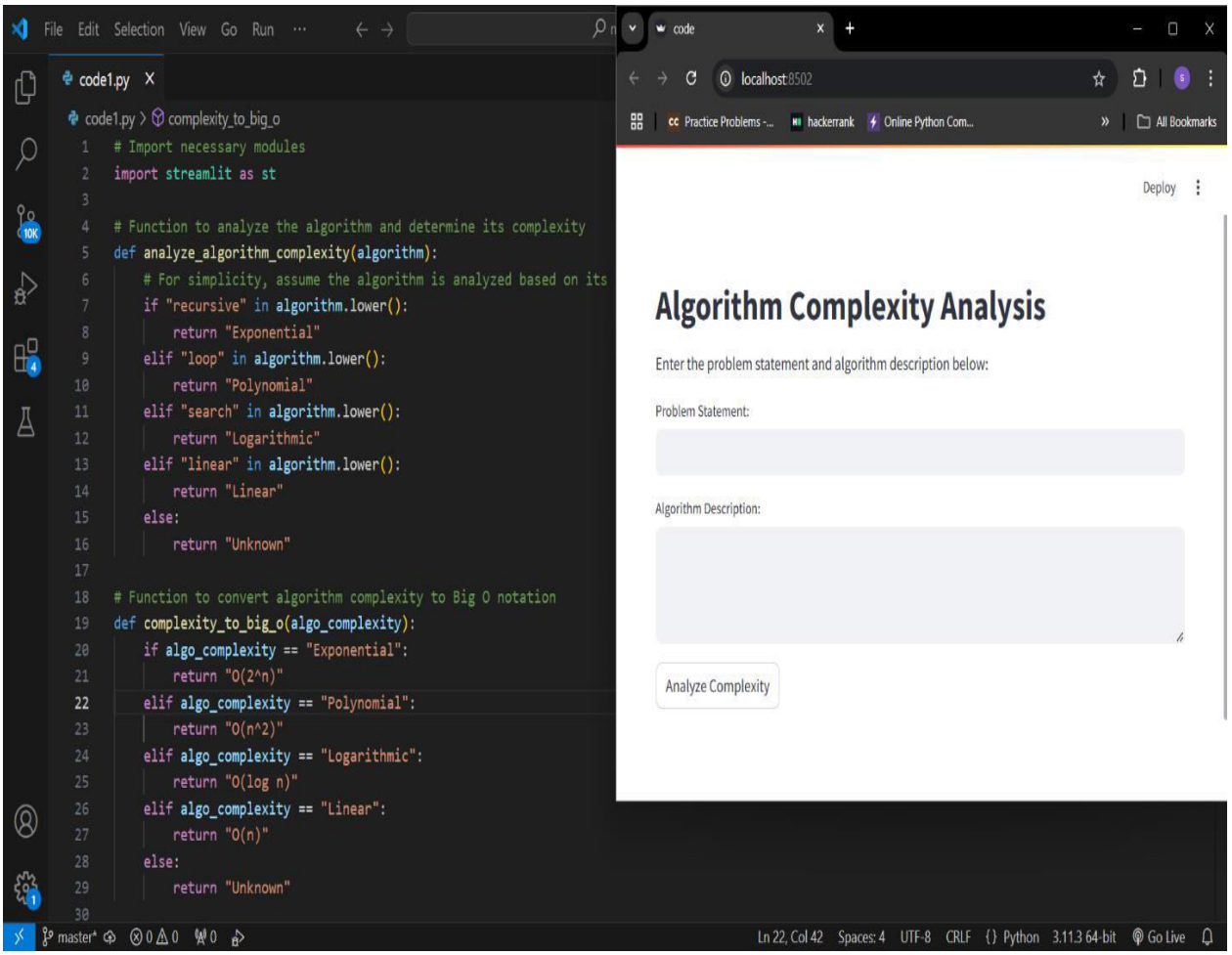
Polynomial Complexity (O(n^k)): Observed in iterative algorithms with nested loops, this class represents growth at polynomial rates.

Logarithmic Complexity (O(log n)): Often present in search algorithms, logarithmic complexity is achieved by reducing the problem size with each operation, as seen in binary search.

Linear Complexity (O(n)): Associated with single-iteration operations, linear complexity is efficient and grows proportionally with input size.

IV. RESULTS AND DISCUSSION

The algorithm complexity analysis tool accurately classified time complexity for standard algorithms, confirming its reliability for basic use cases. Users found the tool intuitive, with real-time feedback and educational value for understanding Big O notation.



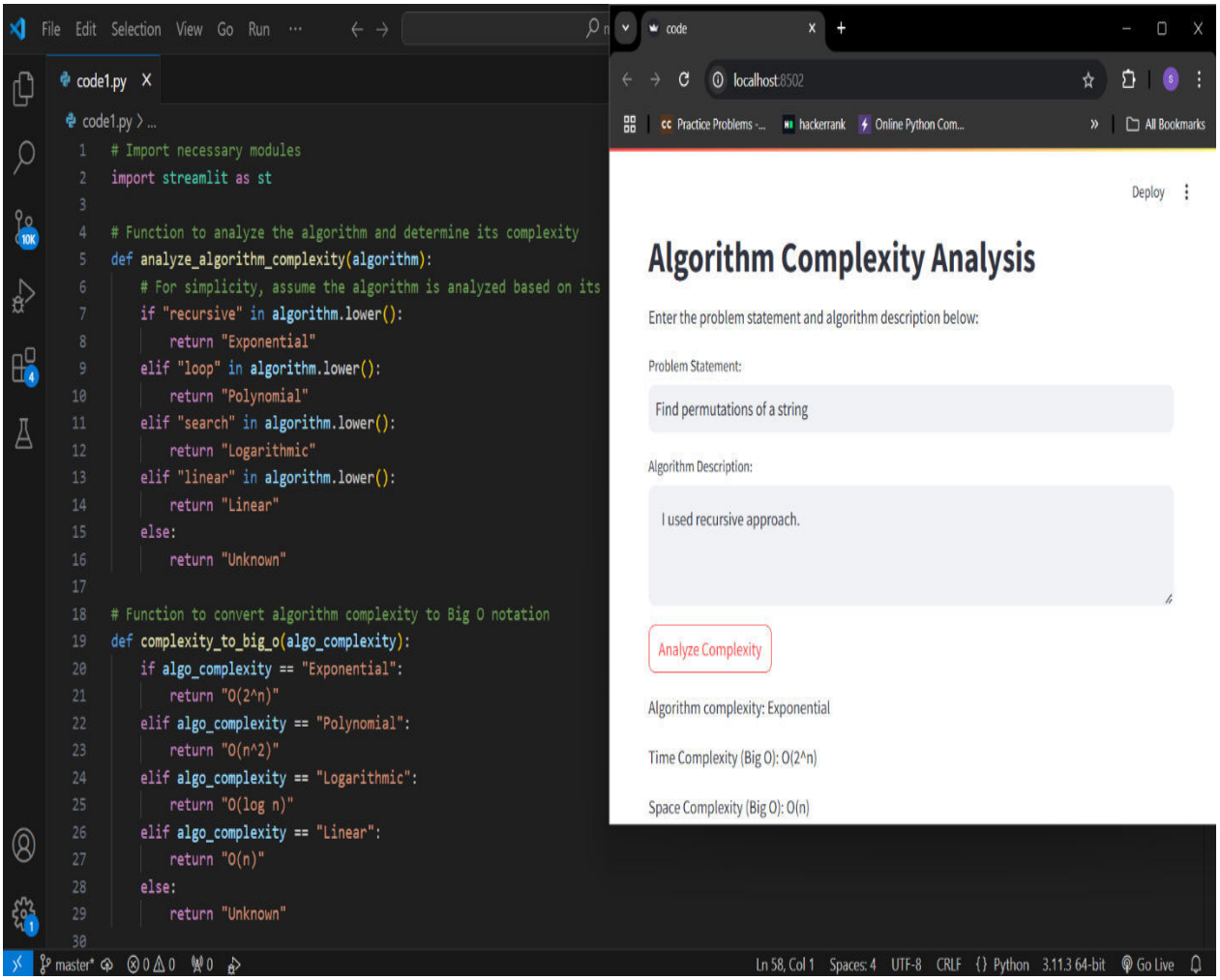
International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.524

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 13, Issue 11, November 2024

|DOI: 10.15680/IJIRSET.2024.1311087|



4.1Preparation of Figures

The figures below represent the operation and the embedding of the various components of the working system of the Complexity Analysis.

4.1.1Formatting Figures

Deployment Diagram: The deployment diagram is used to show the physical arrangement of the software and the hardware used in the project that are deployed. It demonstrates how components are distributed across different nodes and how they interact through potential communication paths. The deployment diagram outlines the architectural setup of the algorithm complexity analysis tool, showcasing the various components, their interactions, and the deployment environment. This tool is a web-based application that uses Streamlit for frontend interaction, hosted on a web server, and relies on a backend to handle complexity classification logic.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.524

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 13, Issue 11, November 2024

|DOI: 10.15680/IJIRSET.2024.1311087|

Components and Architecture:

1.

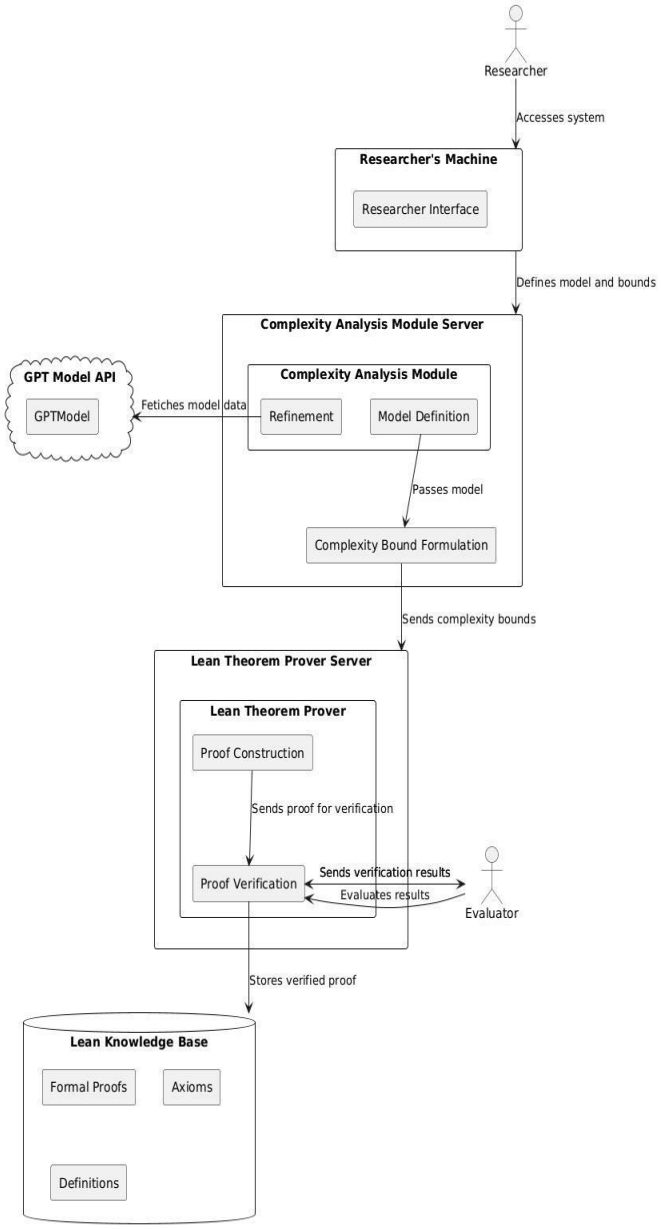


Figure 1: Deployment Diagram

Activity Diagram: It models the workflow of the system or the process while highlighting the sequences of decisions and activities, The diagram helps in visualizing a step-by-step process of the basic working system that the platform follows. The activity diagram outlines the step-by-step flow of operations in the algorithm complexity analysis tool, from user input to final output. This sequence includes user interaction, complexity analysis, result presentation, and optional data storage

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.524

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 13, Issue 11, November 2024

|DOI: 10.15680/IJIRSET.2024.1311087|

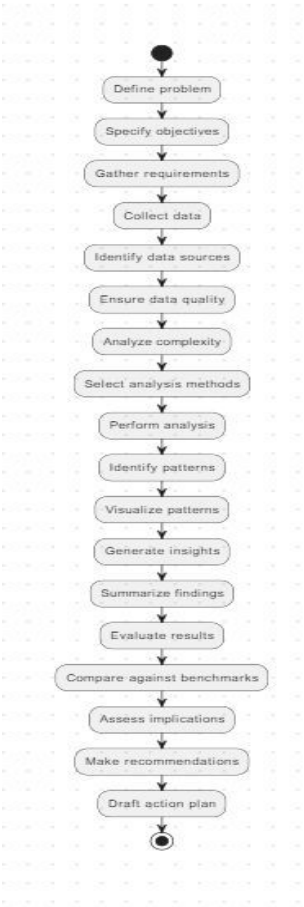


Figure 2: Activity diagram

V. CONCLUSIONS

This study presented a web-based tool for analyzing algorithmic complexity, aimed at simplifying the understanding and classification of algorithm efficiency for developers and students. By leveraging keyword-based detection methods, the tool successfully identifies the complexity class (e.g., linear, polynomial, logarithmic, or exponential) and converts it to Big O notation, providing quick insights into computational performance. The tool's usability and real-time feedback make it an accessible educational resource, bridging theoretical knowledge and practical application in complexity analysis. However, reliance on keywords limits its precision for complex algorithm structures. Future improvements, such as incorporating machine learning models, could enhance the tool's accuracy and adaptability, enabling it to handle a wider variety of algorithms with greater precision.

DECLARATIONS

Study Limitations
None
Funding source None.
Competing Interests
The authors, hereby declare that there are no or competing interests.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.524

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 13, Issue 11, November 2024

|DOI: 10.15680/IJIRSET.2024.1311087|

HUMAN AND ANIMAL RELATED STUDY

The study involves no collection of physical data acquired from humans or animals that are put through rigorous testing nor do they have to give a sample of their fluids or any other bodily secretions or excreta.

Ethical Approval

Provide ethical approval authority name with the reference number. If ethical approval is not required, provide an ethical exemption letter of not required. The author should send scan copy (in pdf) of the ethical approval/exemption letter obtained from IRB/ethical committee or institutional head.

Informed Consent

All of the e-mails made were done with consent as test e-mails provided by the authors themselves.

REFERENCES

[1] Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to Algorithms (3rd ed.). MIT Press. MIT Press: https://mitpress.mit.edu/9780262033848/introduction-to-algorithms-third-edition/

[2] Sedgewick, R., & Wayne, K. (2011). Algorithms (4th ed.). Addison-Wesley Professional. Addison-Wesley: https://www.pearson.com/store/p/algorithms/P100000575062

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INNO SPACE SJIF Scientific Journal Impact Factor

doi cross ref

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

9940 572 462 6381 907 438 ijirset@gmail.com

www.ijirset.com

Scan to save the contact details