Data Structures Visualization

Dr. V.Sireesha1, Dr. Nagaratna P Hegde2, Anisha Kollipara3, Meghana Ganapa4, MSV Sashi Kumar5

1,2,3,4Dept.of CSE, Vasavi College of Engineering, Hyderabad, Telangana, India

5Alight, Hyderabad, Telangana, India

v.sireesha@staff.vce.ac.in, [nagaratnaph@staff.vce.ac.in](mailto:nagaratnaph@staff.vce.ac.in), [anishacse2018@gmail.com,](mailto:anishacse2018@gmail.com) [meghanaganapa@gmail.com](mailto:meghanaganapa@gmail.com), sashi.mamidanna@gmail.com

**Abstract.** Data structures are fundamental for efficiently storing, accessing, and performing operations like insertion, deletion, and data updates. Complex data structures such as Binary Search Trees and Linked Lists can pose challenges when understanding and implementing them without proper visualization tools. The Data Structures Visualization Tool addresses this issue by offering step-by-step visual representations of these operations, facilitating user comprehension of data manipulation. Furthermore, this tool empowers users to create and execute their own custom methods through a simplified programming language designed for various data structures. By utilizing this user-friendly language, users can reduce the hurdles associated with complex syntax and semantics found in traditional programming languages. This, in turn, enables users to concentrate solely on comprehending and effectively utilizing data structures, thereby enhancing their overall understanding and proficiency in efficiently managing data.

**Keywords:** Data Structures, Visualization, Programming Interface, Interactive Education, Python Turtle Graphics.

1. Introduction

Data Structures are one of the most important topics for anyone interested in working in a Computer Science related field. It just so happens that data structures can be incredibly difficult and stressful to understand clearly. This prevents many from effectively using data and performing operations which are of minimum Time and Space Complexity. By visualizing the Data Structures, users can get a clear understanding of what happens, and this can help them better understand their implementation. The Data Structures Visualization Tool can be used by anyone who is familiar with programming and does not require understanding complex syntaxes and structures. The user can easily program their required Data Structures and perform different operations on them. After coding, they can look at the

Visualization of all the Data Structures mentioned, and different operations performed in a step-by-step manner. The user will be able to understand how they work by looking at visual representations instead of the implementation code itself, and by using the simple programming language to create them, they will become more familiar with implementing different Data Structures.

1. Literature Survey

Y. Lavner, R. Cohen, D. Ruinskiy, and H. Ijzerman [1] developed an automatic baby cry detection system using audio signals. They did this using machine learning algorithms like logistic regression classifier and Mel Frequency cepstrum coefficients. S. Asthana, N. Varma, and V. K. Mittal [2] have analyzed the cry patterns of the baby. This dealt with analyzing the audio signals and signal processing methods like autocorrelation and linear prediction analysis of the frequency of baby sounds.

1. Design

The project was implemented using Python’s Turtle module, a tool for producing

interactive GUI (Graphical User Interface) by using “turtles” which can be repositioned and modified to provide different elements of the UI. A list of turtles has been assigned to depict the different elements of a Data Structure, with every turtle being assigned an image to depict on the screen. There are different Classes which encapsulate all the attributes and methods of a Data Structure, with each method manipulating the List of Turtles in such a way that it represents the specific task being performed. Using Python’s Tkinter module, a File Management System has been created which the user can utilize to create new scripts, remove existing scripts, edit scripts, view the contents of a script, and visualize the script. The programs written are converted into tokens using Regular Expressions, and based on the type of token, the required Data Structures are created, and respective methods are called during visualization.

**Fig. 1.** System Design

1. Implementation

The tool has a Programming Interface and a Visualization Window with different data structures such as Array, Stack, Linked List and Binary Search Tree. The implementation of the programming interface and one of the data structures are discussed below.

**A. Implementing the Programming Interface:**

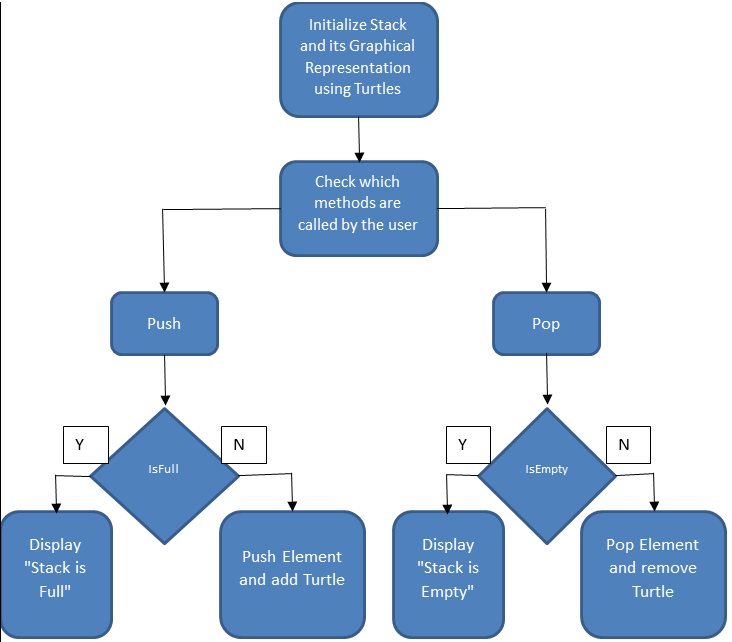
The user is given a command prompt to add, edit, remove, or list all scripts which are stored in the same directory. After selecting a script, the code opens a separate window using the Tkinter library, where users can view and modify the content of the chosen Python script. This editing interface includes a "Save" button to save any changes made to the script. The code also performs error handling, notifying users if the specified script file is not found in the directory.

After saving the script, users can visualize it through the command prompt. The contents of the script are scanned, processed, and interpreted in a structured format. which contains instructions for initializing data structures, performing operations on them, and defining their properties. It reads the input instructions from a file, processes them sequentially, and organizes the information into relevant data structures and operations. Initially, the code extracts and categorizes instructions into three main sections: declaration, initialization, and code execution. In the declaration section, the code identifies and extracts details about data structures, including their names, types, and capacities. The initialization section assigns initial values to objects of these data structures. The code then analyzes the code execution section, identifying method calls and operations to be performed on the initialized data structures. It organizes these operations by associating them with the respective data structures and methods. Subsequently, the code consolidates all this information into a dictionary, where data structures are linked to their properties, method calls, and associated parameters. Finally, the code passes this data to a function, along with a list of objects, for visualization.

**Fig. 2.** Implementation of Programming Interface

**B. Implementing the Stack Data Structure:**

A Python class is created to visually depict individual stack elements using the Turtle graphics library. Each "Stack Turtle" instance represents one stack item, with parameters for its position (x and y coordinates) and associated data. Each instance includes graphical elements, such as a Turtle graphic resembling a stack icon and a pen for displaying text data above it. The class offers methods to relocate the graphical representation, write, clear, and hide data, set new data, and display data when clicked. Essentially, it provides a graphical representation for stack elements.



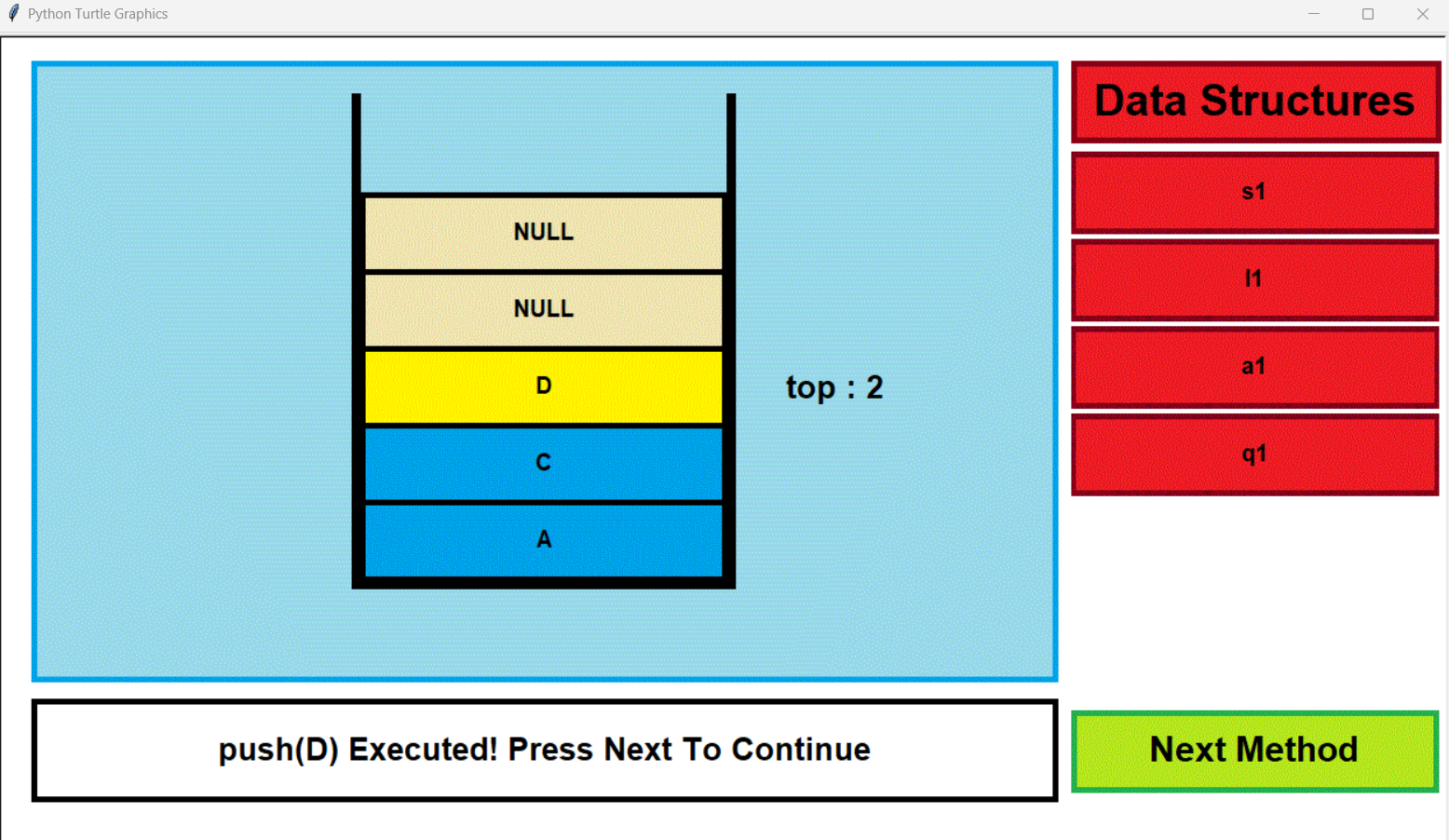
**Fig. 3.** Implementation of Stack Data Structure

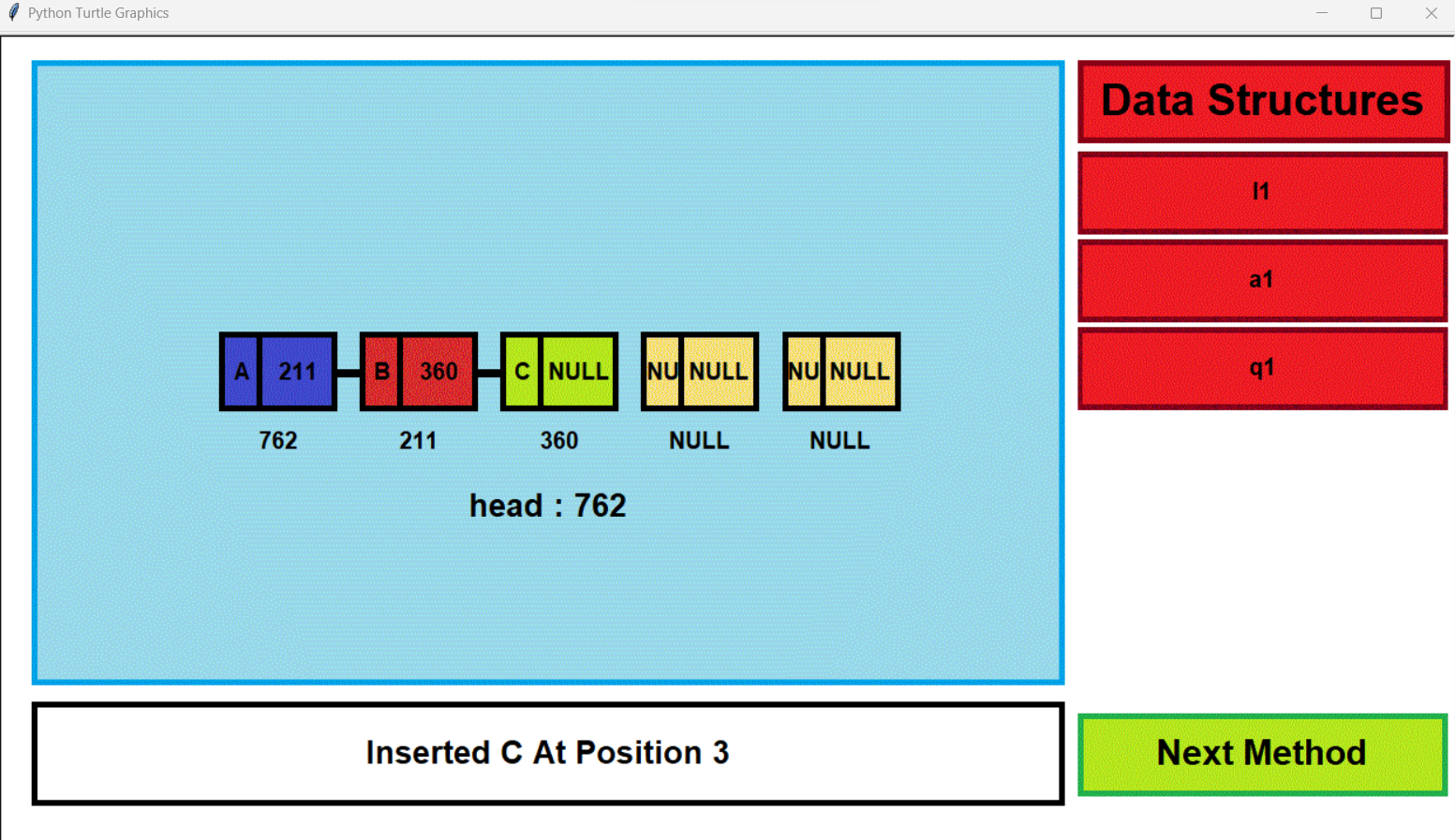
**5 Results**

The Visualization of different Data Structures has been achieved along with the

implementation of a Programmable Interface to program different Structures and

call their respective methods.

 **Fig. 4.** Visualization of Stack Data Structure

 **Fig. 5.** Visualization of Linked List Data Structure.

The above images represent some Data Structures that have been implemented in the Visualization Tool, with the representation of certain attributes of the Structure, such as “top” in the case of a Stack. A list of declared Data Structure objects is depicted on the right along with a “Next” button to continue the Visualization. There is a Text Box at the bottom which informs the user about the operation being performed and the various sub-operations required to perform them.

**6 Advantages**

* No Internet Dependency i.e., Functionality offline.
* Improved application responsiveness due to local processing.
* Users can use custom input of code and use according to their needs.
* Users can save and share their scripts
* Support for various data types such as String and Float

1. 7 Conclusion
2. The tool we created accurately represents the mentioned Data Structures along with their operations in a simple graphical format. All the methods work as expected and handle exceptions as well. The Structures appear in the sequence in which they were declared, and the user can move through every Structure and its methods in sequential manner. Any operation being performed is described in a text box along with sub-operations such as comparisons and checks to make it user-friendly. Every template of data has been assigned a random color whenever it is given a value which makes the tool more attractive. The Programmable Interface can compile all the methods which have been implemented and makes the tool more interactive so that the user can code their own programs.
3. 8 Future Work

The project's purpose was to develop a Visualization tool for simple visualizations.

It currently only works on String data types and does comparisons based on ASCII

values. Other primitive data types such as Integer, Float and Boolean can be added

to make the tool more diverse. It can be extended to implement other complicated

Data Structures such as Red-Black Trees, AVL Trees, and B-Trees. Extensive Exception Handling can be implemented in the Programmable Interface to make the programming experience much more user-friendly.

1. Acknowledgement
2. We thank Vasavi College of Engineering (Autonomous), Hyderabad for the support extended towards this work.
3. References

[1] https://docs.python.org/3/library/turtle.html [Accessed: 26-08-23].

[2] https://docs.python.org/3/library/tkinter.html [Accessed: 26-08-23].

[3] https://www.geeksforgeeks.org/array-data-structure/ [Accessed: 26-08-23].

[4] https://www.geeksforgeeks.org/stack-data-structure/ [Accessed: 26-08-23].

[5] https://www.geeksforgeeks.org/queue-data-structure/ [Accessed: 26-08-23].

[6] <https://www.geeksforgeeks.org/data-structures/linked-list> [Accessed: 26-08-23].

[7] <https://www.geeksforgeeks.org/binary-tree-data-structure/>

[Accessed: 26-08-23].

[8] <https://www.geeksforgeeks.org/binary-search-tree-data-structure/>

[Accessed: 26-08-23].