A Minor Project report on

DATA VISUALIZATION OF RED BLACK TREE

A Dissertation submitted to JNTU Hyderabad in partial fulfilment of the academic requirements for the award of the degree.

Bachelor of Technology

in

Computer Science and Engineering

Submitted by

G.Mahendra (21H55A0504)
H.Laxman (21H55A0508)
M.Jyoshna (21H55A0513)

Under the esteemed guidance of
MS.J.Ranjith
(Assistant Professor)

EXPLORE TO INVENT

Department of Computer Science and Engineering

CMR COLLEGE OF ENGINEERING & TECHNOLOGY
(UGC Autonomous)
*Approved by AICTE *Affiliated to JNTUH *NAAC Accredited with A^+ Grade

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD - 501401.

2020- 2024

CMR COLLEGE OF ENGINEERING & TECHNOLOGY

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD – 501401

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to clarify that the Mini Project II report entitled“**DATA VISUALIZATION OF RED BLACK TREE**” being submitted by G.Mahendra(21H55A0504),H.Laxman(21H55A0508),M.Jyoshna(21H55A0513) in partial fulfilment for the award of **Bachelor of Technology in Computer Science and Engineering** is a record of bonafide work carried out his/her under my guidance and supervision.

The results embodies in this project report have not been submitted to any other University or Institute for the award of any Degree.

Mr.J.Ranjith
Assistant Professor
Dept.of CSE

Dr.Siva Skanda Sanagala
Associate Professor and HOD
Dept.of CSE

ACKNOWLEDGEMENT

With great pleasure we want to take this opportunity to express my heartfelt gratitude to all the people who helped in making this project work a grand success.

We are greatful to **Mr.J.Ranjith,Associate Professor**, Department of Computer Science and Engineering for his valuable technical suggestions and guidance during the execution of this project work.

We would like to thank **Dr. Siva Skandha Sanagala,** Head of the Department of Computer Science and Engineering, CMR College of Engineering and Technology, who is the major driving forces to complete my project work successfully.

We are very grateful to **Dr. Vijaya Kumar Koppula**, Dean-Academics, CMR College of Engineering and Technology, for his constant support and motivation in carrying out the project work successfully.

We are highly indebted to **Major Dr. V A Narayana,** Principal, CMR College of Engineering and Technology, for giving permission to carry out this project in a successful and fruitful way.

We would like to thank the **Teaching & Non- teaching** staff of Department of Computer Science and Engineering for their co-operation.

We express our sincere thanks to **Shri. Ch. Gopal Reddy**, Secretary, CMR Group of Institutions, for his continuous care.

Finally, We extend thanks to our parents who stood behind us at different stages of this Project. We sincerely acknowledge and thank all those who gave support directly and indirectly in completion of this project work.

G.Mahendra	(21H55A0504)
H.Laxman	(21H55A0508)
M.Jyoshna	(21H55A0513)

DECLARATION

We hereby declare that results embodied in this Report of Project on "DATA VISUALIZATION OF RED BLACK TRTEE" are from work carried out by using partial fulfilment of the requirements for the award of B. Tech degree. We have not submitted this report to any other university/institute for the award of any other degree.

NAME	ROLL NO	SIGNATURE
G.MAHENDRA	(21H55A0504)	
H.LAXMAN	(21H55A0508)	
M.JYOSHNA	(21H55A0513)	

TABLE OF CONTENTS

ABSTRACT

Red-Black Trees are fundamental data structure known for their self-balancing properties, efficent operations, and applications in various fields. As it aids in comprehending their structure, analyzing algorithmic behaviour, and implementing efficient solutions. This provides an overview of visualization techniques and tools,emphasizing their role in enhancing understanding and facilitating project development.Using a red-black, viewers can quickly identify key data points and make informed decisions based on information presented. Researchers were divided in two separated communities, AVL supporters and Red-Black ones. Indeed, AVL trees are commonly used for retrieval applications whereas Red Black trees are used in updates operations, so, the choice of a structure must be done firstly even if the operations are not known to be searches or updates. That is the main reason why we propose a common tree with the same complexity and memory space, representing both an AVL and a Red-Black tree, this new tree allows to gather together the two communities on one hand, and to expand the scope of AVL and Red-Black tree applications on the other hand. These conditions guarantee that the longest path from root to leaf can be no more than twice the shortest (the only difference being individual red nodes interspersed along the way), so the penalty of locating an element in a red-black tree is minor relative to that in a perfectly-balanced tree. Okasaki properly introduced red-black trees into the functional world when he gave a concise, elegant method of element insertion. To account for the local property violation, the tree is re-balanced as the traversal recedes. This operation rearranges trees of one of the forms.

CHAPTER 1

INTRODUCTION

Binary search trees are very efficient for many applications in computer science but have poor worst-case performance. We can make the case that when a tree is perfect balanced it takes a long time to traverse down it because it has height at most Log (n) , unfortunately keeping a perfect balance of a tree is rude and so expensive in practice, that's why balanced trees are introduced, where costs are guaranteed to be logarithmic while ensuring that the tree remains almost balanced, examples of these trees are: AVL trees and Red-Black trees. Both AVL trees and Red-Black trees are very popular data structures. They are the most used self-balancing binary search trees. Indeed, they are implemented and integrated in many programming languages like JAVA and C++. They are also used generally to implement dictionaries and associative arrays. After the invention of AVL trees, other propositions are made in order to improve performance or to give simplify algorithms: Foster gives complements studies expanded to the original AVL tree, the second improved AVL tree is the generalization of AVL trees which allows unbalances up to a small integer. Another main structure is a one-sided height balanced tree (OSHB) which makes a restriction on heights of node's sons so that the right son never has smaller height than the left one, insertion and deletion algorithms are in O (log2 n) time, after that optimum algorithms for OSHB tree are proposed in O (Log n) time with more complicated algorithms than the original AVL trees. The original data structure of Red Black trees is invented in 1972 by Rudolf Bayer under the name: "Symmetric Binary B-trees", a few years after, a new form of the original structure is proposed where tree balance is expressed Lynda Bounif, Djamel Eddine Zegour LCSI Laboratory, National School of computer science ESI (ex INI) Algeria. Using red and black colors, this structure is difficult to understand and implement, therefore many works appeared in order to simplify the algorithms, AA tree is a powerful simplification of RedBlack trees with the same performance and much more simplicity and simple

Recently the majority of works in terms of AVL and Red-Black trees aims basically to simplify rather than anything else, introduces a new simpler insertion and deletion algorithms for AVL tree by using virtual nodes and a brief study of AVL trees using this concept is presented in , then gives a new algorithms and shows how easily maintain the balance factor after an updating operation. When it comes to Red-Black trees a revisited version has been proposed where the code is considerably reduced compared to the implementation proposed in where a complete C# implementation of Red-Black trees is presented including all the implementation details for insertion and deletion.

1.1. PROPOSED SYSTEM

The red-Black tree is a binary search tree. The prerequisite of the red-black tree is that we should know about the binary search tree. In a binary search tree, the values of the nodes in the left subtree should be less than the value of the root node, and the values of the nodes in the right subtree should be greater than the value of the root node. Each node in the Red-black tree contains an extra bit that represents a color to ensure that the tree is balanced during any operations performed on the tree like insertion, deletion, etc. In a binary search tree, the searching, insertion and deletion take O(log2n) time in the average case, O(1) in the best case and O(n) in the worst case.

1.2 ADVANTAGES OF PROPOSED SYSTEM

It is a self-balancing Binary Search tree. Here, self-balancing means that it balances the tree itself by either doing the rotations or recoloring the nodes. This tree data structure is named as a Red-Black tree as each node is either Red or Black in color. Every node stores one extra information known as a bit that represents the color of the node. For example, 0 bit denotes the black color while 1 bit denotes the red color of the node. Other information stored by the node is similar to the binary tree, i.e., data part, left pointer and right pointer. In the Red-Black tree, the root node is always black in color. In a binary tree, we consider those nodes as the leaf which have no child. In contrast, in the Red-Black tree, the nodes that have no child are considered the internal nodes and these nodes are connected to the NIL nodes that are always black in color. The NIL nodes are the leaf nodes in the Red-Black tree. If the node is Red, then its children should be in Black color. In other words, we can say that there should be no redred parent-child relationship. Every path from a node to any of its descendant's NIL node should have same number of black nodes.

1.3 SYSTEM EXISTING

It is the first balanced binary data structure. It's named after its two inventors, G.M. Adelson-Velskii and E.M. Landis. In an AVL tree, the heights of the two child sub-trees of any node differ by at most one, so each node alone in the tree represents an AVL tree, A balance factor is then added in each node in order to maintain the balance of the tree, it can takes only the values: 0, -1 or +1, so the balancing actions have to be done when the balance factor becomes 2 or -2.

1.4 DISADVANTAGES OF EXISTING SYSTEM

Several comparison studies of balanced trees especially for AVL and Red-Black tree are available; the most important one is, it takes in considerations these two factors: the height of the tree per operation and the number of rotations, and varies the rate of search, insertion and deletion operation, the results of these studies, shows that AVL tree is more efficient that Red-Black tree, especially in lookup-intensive applications, because AVL tree is more balanced than Red-Black trees, It has a height bounded by 1.44 Log (n) where n is the number of nodes in the tree and log the base-two algorithm.

CHAPTER 2

BACKGROUND WORK

2.1 INTRODUCTION

Testing is the debugging program is one of the most critical aspects of the computer programming triggers, without programming that works, the system would never produce an output of which it was designed. Testing is best performed when user development is asked to assist in identifying all errors and bugs. The sample data are used for testing. It is not quantity but quality of the data used the matters of testing. Testing is aimed at ensuring that the system was accurately an efficiently before live operation commands.

Testing objectives

The main objective of testing is to uncover a host of errors, systematically and with minimum effort and time. Stating formally, we can say, testing is a process of executing a program with intent of finding an error. User Interface Design is concerned with how users add information to the system and with how the system presents information back to them. Data Design is concerned with how the data is represented and stored within the system. Finally, Process Design is concerned with how data moves through the system, and with how and where it is validated, secured and/or transformed as it flows into, through and out of the system. At the end of the systems design phase, documentation describing the three sub-tasks is produced and made available for use in the next phase.

1 A successful test is one that uncovers an as yet undiscovered error.

2 A good test case is one that has probability of finding an error, if it exists.

3 The test is inadequate to detect possibly present errors.

4 The software more or less confirms to the quality and reliable standards.

2.2 Levels of Testing

Code Testing

This examines the logic of the program. For example, the logic for updating various sample data and with the sample files and directories were tested and verified.

Specification Testing

Executing this specification starting what the program should do and how it should performed under various conditions. Test cases for various situation and combination of conditions in all the modules are tested.

Unit testing

In the unit testing we test each module individually and integrate with the overall system. Unit testing focuses verification efforts on the smallest unit of software design in the module. This is also known as module testing. The module of the system is tested separately. This testing is carried out during programming stage itself. In the testing step each module is found to work satisfactorily as regard to expected output from the module. There are some validation checks for fields also. For example the validation check is done for varying the user input given by the user which validity of the data entered. It is very easy to find error debut the system.

Each Module can be tested using the following two Strategies:

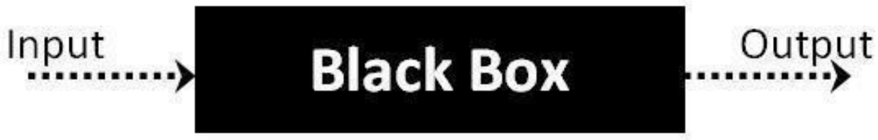
1. Black Box Testing
2. White Box Testing

BLACK BOX TESTING

What is Black Box Testing?

Black box testing is a software testing techniques in which functionality of the software under test (SUT) is tested without looking at the internal code structure, implementation details and knowledge of internal paths of the software. This type of testing is based entirely on the software requirements and specifications.

In Black Box Testing we just focus on inputs and output of the software system without bothering about internal knowledge of the software program.



The above Black Box can be any software system you want to test. For example : an operating system like Windows, a website like Google ,a database like Oracle or even your own custom application. Under Black Box Testing , you can test these applications by just focusing on the inputs and outputs without knowing their internal code implementation.

WHITE BOX TESTING

White Box Testing is the testing of a software solution's internal coding and infrastructure. It focuses primarily on strengthening security, the flow of inputs and outputs through the application, and improving design and usability. White box testing is also known as clear, open, structural, and glass box testing.

It is one of two parts of the "box testing" approach of software testing. Its counter-part, black box testing, involves testing from an external or end-user type perspective. On the other hand, Whitebox testing is based on the inner workings of an application and revolves around internal testing. The term "white box" was used because of the seethrough box concept. The clear box or white box name symbolizes the ability to see through the software's outer shell (or "box") into its inner workings. Likewise, the "black box" in "black box testing" symbolizes not being able to see the inner workings of the software so that only the end-user experience can be tested.

STEP 1) UNDERSTAND THE SOURCE CODE

The first thing a tester will often do is learn and understand the source code of the application. Since white box testing involves the testing of the inner workings of an application, the tester must be very knowledgeable in the programming languages used in the applications they are testing. Also, the testing person must be highly aware of secure coding practices. Security is often one of the primary objectives of testing software. The tester should be able to find security issues and prevent attacks from hackers and naive users who might inject malicious code into the application either knowingly or unknowingly.

STEP 2) CREATE TEST CASES AND EXECUTE

The second basic step to white box testing involves testing the application's source code for proper flow and structure. One way is by writing more code to test the application's source code. The tester will develop little tests for each process or series of processes in the application. This method requires that the tester must have intimate knowledge of the code and is often done by the developer. Other methods include manual testing, trial and error testing and the use of testing tools as we will explain further on in this article.

System Testing

Once the individual module testing is completed, modules are assembled and integrated to perform as a system. The top down testing, which began from upper level to lower level module, was carried out to check whether the entire system is performing satisfactorily.

There are three main kinds of System testing:

1 Alpha Testing

2 Beta Testing

3 3 Acceptance Testing

Alpha Testing

This refers to the system testing that is carried out by the test team with the Organization.

Beta Testing

This refers to the system testing that is performed by a selected group of friendly customers.

Acceptance Testing

This refers to the system testing that is performed by the customer to determine whether or not to accept the delivery of the system.

Integration Testing

Data can be lost across an interface, one module can have an adverse effort on the other sub functions, when combined, may not produce the desired major functions. Integrated testing is the systematic testing for constructing the uncover errors within the interface. The testing was done with sample data. The developed system has run successfully for this sample data. The need for integrated test is to find the overall system performance.

Output testing

After performance of the validation testing, the next step is output testing. The output displayed or generated by the system under consideration is tested by asking the user about the format required by system. The output format on the screen is found to be correct as format was designed in the system phase according to the user needs.

Hence the output testing does not result in any correction in the system.

Test plan

The test-plan is basically a list of testcases that need to be run on the system. Some of the testcases can be run independently for some components (report generation from the database, for example, can be tested independently) and some of the testcases require the whole system to be ready for their execution. It is better to test each component as and when it is ready before integrating the components. It is important to note that the testcases cover all the aspects of the system (ie, all the requirements stated in the RS document).

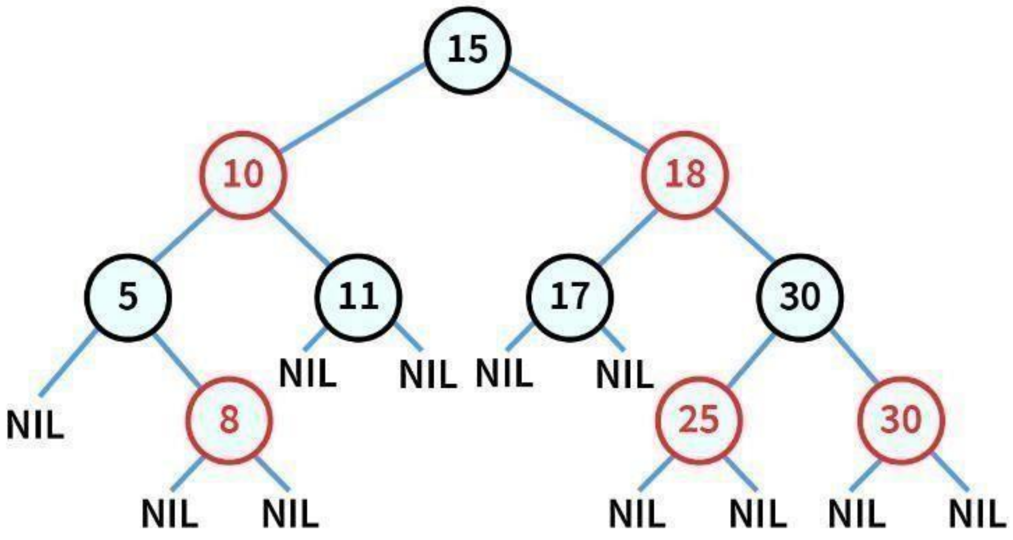
CHAPTER 3

PROPOSED SYSTEM

3.1 SYSTEM ARCHITECTURE

Architecture Flow

Below architecture diagram represents mainly flow of request from the users to database through servers. In this scenario overall system is designed in three tiers separately using three layers called presentation layer, business layer, data link layer. This project was developed using 3-tier architecture.



3.2 3-Tier Architecture

The three-tier software architecture (a three layer architecture) emerged in the 1990s to overcome the limitations of the two-tier architecture. The third tier (middle tier server) is between the user interface (client) and the data management (server) components. This middle tier provides process management where business logic and rules are executed and can accommodate hundreds of users (as compared to only 100 users with the two tier architecture) by providing functions such as queuing, application execution, and database staging.

The three tier architecture is used when an effective distributed client/server design is needed that provides (when compared to the two tier) increased performance, flexibility, maintainability, reusability, and scalability, while hiding the complexity of distributed processing from the user. These characteristics have made three layer architectures a popular choice for Internet applications and net-centric information systems.

Advantages of Three-Tier

- Separates functionality from presentation.
- Clear separation – better understanding.
- Changes limited to well define components.
- Can be running on WWW.
- Effective network performance.

SYSTEM DESIGN

The System Design Document describes the system requirements, operating environment, system and subsystem architecture, files and database design, input formats, output layouts, human-machine interfaces, detailed design, processing logic, and external interfaces.

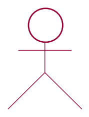
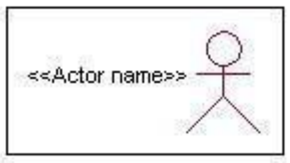
3.3 UML DIAGRAMS

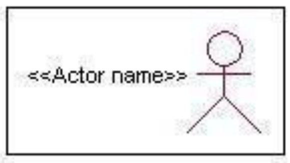
Global Use Case Diagrams

Identification of actors

Actor: Actor represents the role a user plays with respect to the system. An actor interacts with, but has no control over the use cases.

Graphical representation



An actor is someone or something that:

- Interacts with or uses the system.
- Provides input to and receives information from the system.
- Is external to the system and has no control over the use cases.

Actors are discovered by examining:

- Who directly uses the system?
- Who is responsible for maintaining the system?
- External hardware used by the system.
- Other systems that need to interact with the system.

3.4 Flow of Events

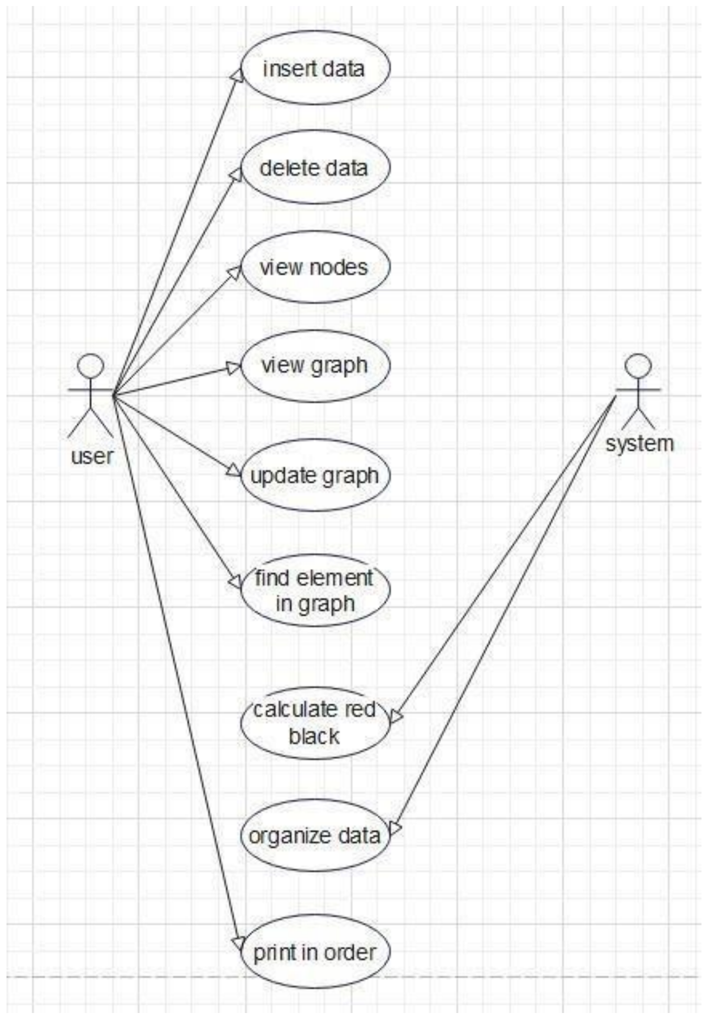
A flow of events is a sequence of transactions (or events) performed by the system. They typically contain very detailed information, written in terms of what the system should do, not how the system accomplishes the task. Flow of events are created as separate files or documents in your favorite text editor and then attached or linked to a use case using the Files tab of a model element.

A flow of events should include:

- When and how the use case starts and ends
- Use case/actor interactions
- Data needed by the use case
- Normal sequence of events for the use case
- Alternate or exceptional flows

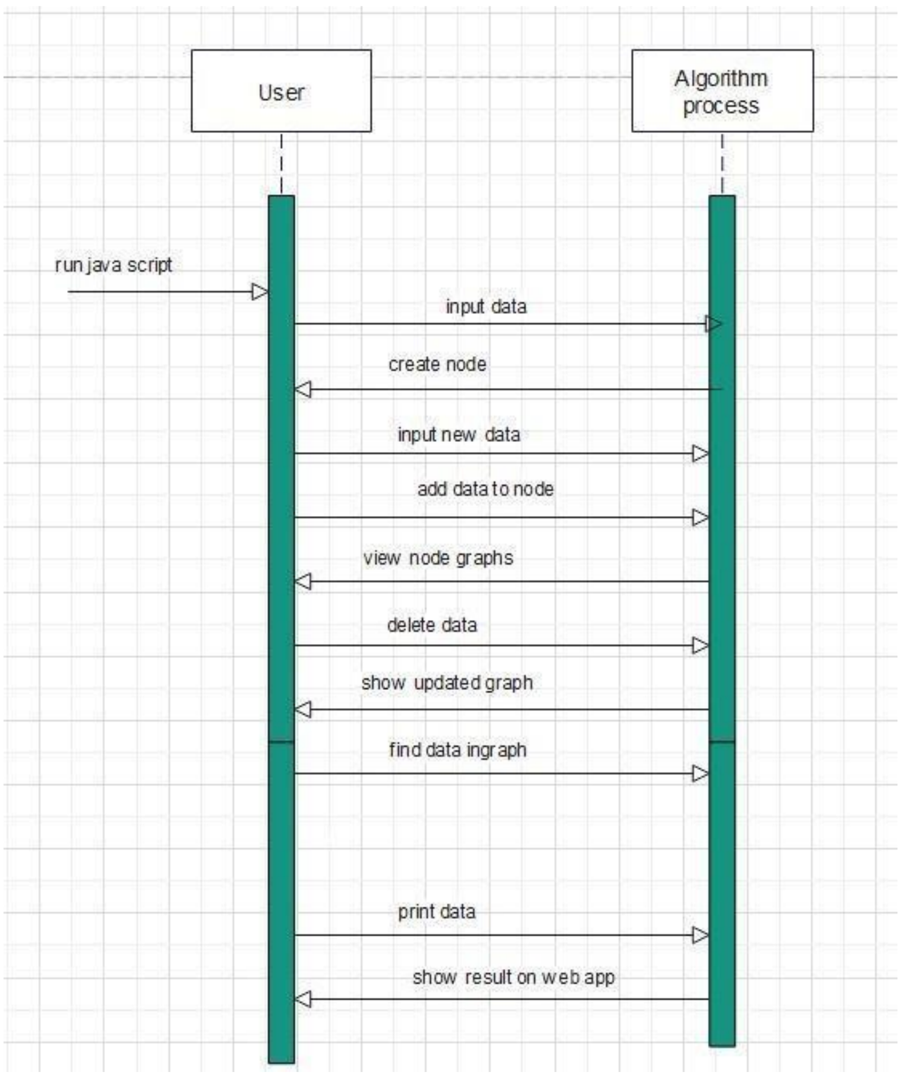
3.5 Construction of Use case diagrams

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



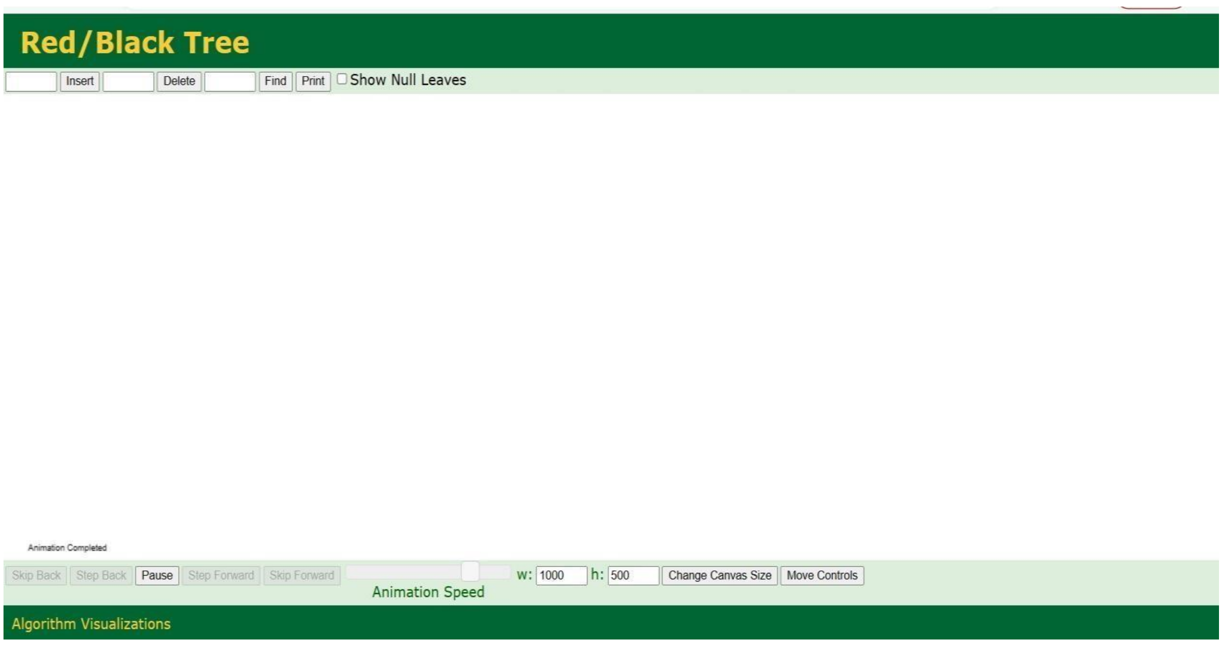
3.6 SEQUENCE DIAGRAMS

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

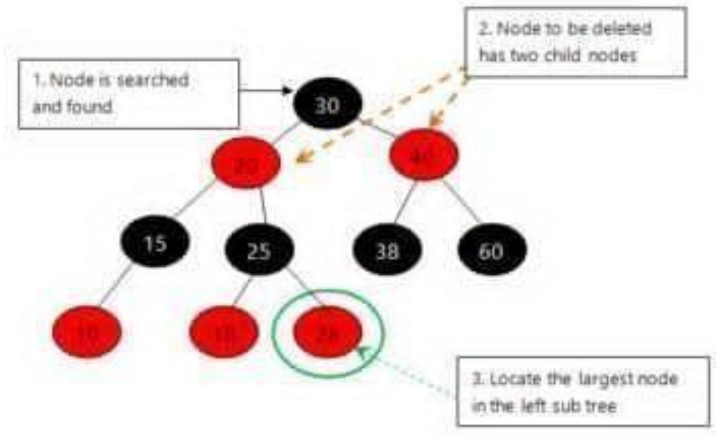


CHAPTER 4

RESULTS

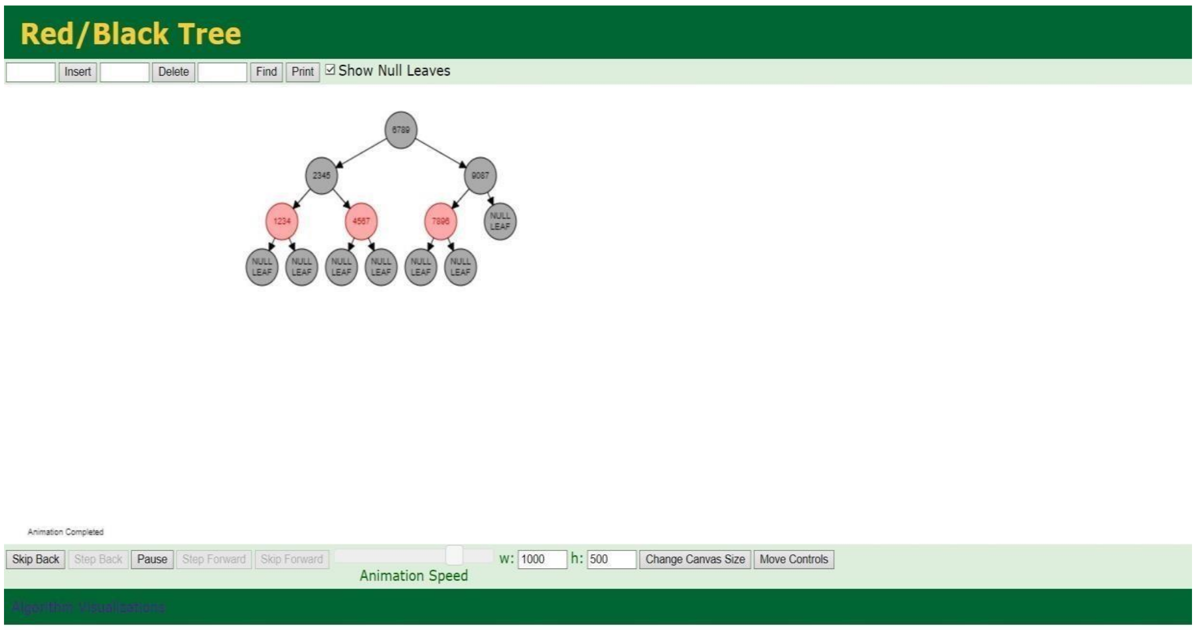


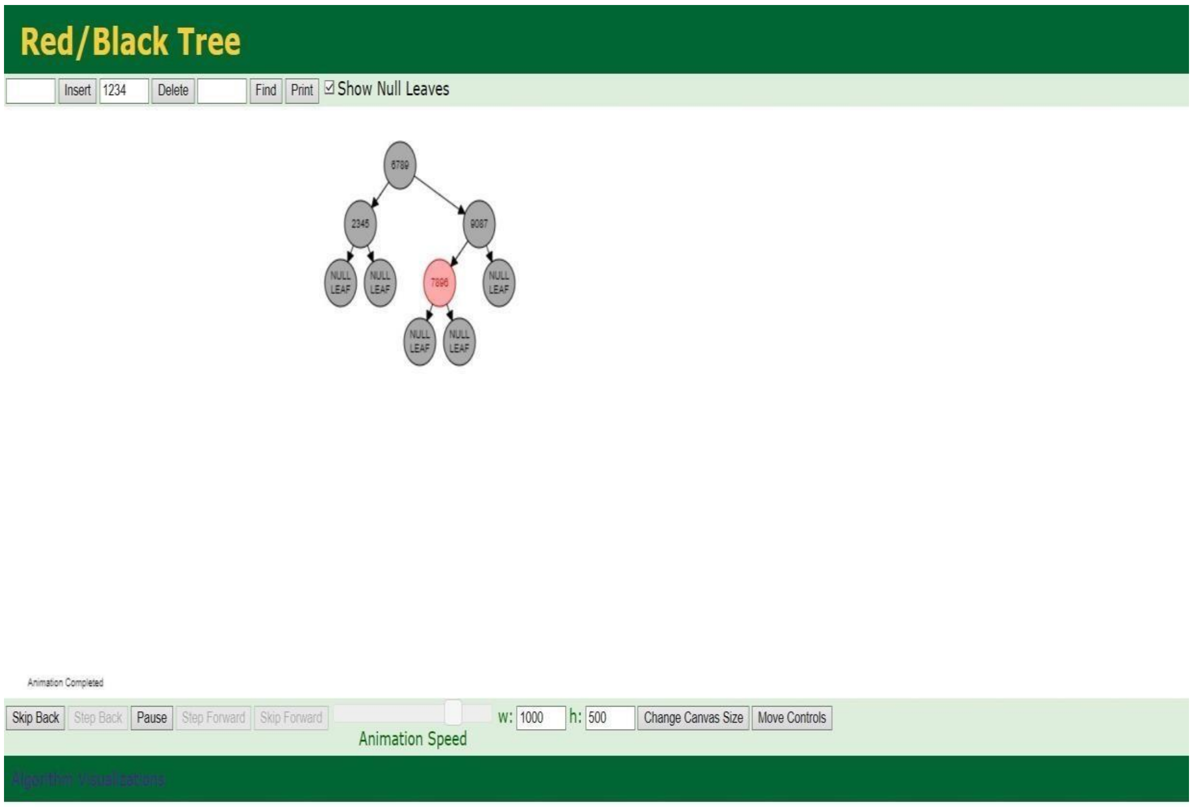
Because of the rules of the Red-Black Tres, its height is at most 2log(N+1).


1. Node is searched and found
2. Node to be deleted has two child nodes
30
15
25
38
60
3. Locate the largest node in the left sub tree

Analysis:

1. **Finding node.**
2. **Inserting node.**
3. **Deleting node.**


Red/Black Tree
Insert
Delete
Find
Print
☑ Show Null Leaves
6789
2345
9087
1234
4567
7896
NULL LEAF
Animation Completed
Skip Back
Step Back
Pause
Step Forward
Skip Forward
Animation Speed
w: 1000
h: 500
Change Canvas Size
Move Controls
Algorithm Visualizations


Red/Black Tree
Insert
1234
Delete
Find
Print
☑ Show Null Leaves
6789
2345
9087
7896
NULL LEAF
Animation Completed
Skip Back
Step Back
Pause
Step Forward
Skip Forward
Animation Speed
w: 1000
h: 500
Change Canvas Size
Move Controls
Algorithm Visualizations

CHAPTER 5

CONCLUSION

5.1 Conclusion

Red-black tree needs an extra bit on each node, compar ed to classic binary search tree, to record the node's color, but it is more efficient on performing all operations than the classic binary search tree does due to its balancing property. Red-black tree is better at insertion and deletion than AVL tree owing to its fewer rotations; lookup in red-black tree is slower than that of AVL tree because that AVL tree maintains a more rigid balance than red-black tree does. We can still find red-black trees widely implemented in many libraries, projects, etc., which implies that it has a good performance in practice. In a sense we can say that red-black tree is one of the most efficient data structure in computer science.

5.2 Future Enhancement

- By implementing these future work suggestions, the data visualization of Red-Black trees can become more interactive, informative, and accessible, facilitating a better understanding of this fundamental data structure.

- In future we provide customization options to users, allowing them to change the visualization settings according to their preferences. This may include options to adjust the size and shape of nodes, modify the color scheme, or toggle the display of certain elements.

CHAPTER 6

REFERENCES

Robert. Sedgewick and Addison. Wesley. “Algorithms in Java, Parts 1-4. Professional. 768 pages”. 23 juil. 2002.

[2] AdelsonVelskii, M., and Evgenii Mikhailovich Landis. “An algorithm for the organization of information”. Dokl. Akad. Nauk SSSR 146, pp. 263-266. English translation in Soviet Math. Dokl. 3, pp. 1259-1262, 1962.

[3] Foster, Caxton C. "Information retrieval: information storage and retrieval using AVL trees." Proceedings of the 1965 20th national conference. ACM, 1965.

[4] Foster, Caxton C. "A generalization of AVL trees." Communications of the ACM 16.8 (1973): 513-517.

[5] Hirschberg, Daniel S. "An insertion technique for one-sided height-balanced trees." Communications of the ACM 19.8 (1976): 471-473.