A NOVEL APPROACH FOR CONSTRUCTING BINARY TREE FROM INORDER TRAVERSAL

Teshome Fenta , Lamesginew Andargie , Gizachew Melkamu

Abstract— A data structure is helpful for storing and organizing data within a computer system so that it can be accessed and worked with in appropriate ways. One of a data structure is a tree, a graph in which every node (except root) has only one parent. One application of a binary tree, a tree when each node has at most two child, is to develop a binary expression tree for mathematical expressions. In this paper, we develop a new algorithm which converts directly a given mathematical expression (infix form) to a binary expression tree without intermediate prefix or postfix forms. We also try to show how this algorithms works using example.

Index Terms— binary expression tree, binary tree, tree traversal

1 INTRODUCTION

In the field of computer science, there is basic thing called data which is going to be stored in the computer system. Group of the same or different data elements that can have the same or different size, also called members, together under common name is the main concern of data structure. Hence, storing and organizing this data in a certain manner in a data structure helps us to manage and use it efficiently within the computer system. Items found in a data structure are stored in memory, and to manipulate these data by the software, different operations are provided like add, insert remove and others. Between and/or among data items in a data structure, there is a relationship.

In addition, data structures can provide means for efficient management of large sized data, like large databases. Furthermore, efficient algorithms are designed from efficient data structures. In a software design, instead of emphasizing on the design methods of algorithms, they are focusing on design methods of efficient data structures [2].

There are different data structure types described in different literature. To mention some of them, are arrays, linked lists, stacks, queues, graphs, trees and others. As stated in [1, 5], set of same type of objects arranged nearby in a computer memory can be taken as array. Composite data consisting many similar and individual items like list of names, bank transactions, city temperature values and others are best candidates to be represented by arrays. Each object constituting the array can be accessed by the position within the array called index.

Accordingly, a linked list is one where each data item points to its neighbors. Like array, in list also there are different objects constituting the list. But, in contrary to array, each element within the list has its own memory block, but in array one block of memory is allocated for all elements of the array. Linked lists are made up of nodes, and each node in a list contains a reference to the next node [1].

The other data structures are stacks and queues. As [10] states that, a stack is a data structure in which operations like storing and retrieving data is performed in only its top end. The last element inserted to a stack is the first it is to be removed, and hence stack is a LIFO (Last In First Out) structure. Unlike

- *Teshome Fenta Bitew received his B.Sc. Degree in Information Studies from Jimma University, Ethiopia in 2010 and his M.Sc. in Information System in 2015, form the School of Electrical Engineering and Computing from Adama Science and Technology University, Ethiopia. He is now a Lecturer in Debre Markos University, Ethiopia since 2012. His main research interests are Image Processing, Machine Learning, Data Science, Security and Privacy, Natural Language Processing and Cloud Computing. He is a member of professional society like Ethiopian Space Science Society (ESSS).E-mail: algewg@gmail.com*
- *Lamesginew Andargie Alamirew received his B.Sc. degree in Computer Science & IT from Adama Science and Technology University in 2009, and his M.Sc. in Computer Science Bahir Dar Institute of Technology in 2015, Ethiopia. His main research interest are Data Structures and Algorithms, Big data analytics, Machine learning, Deep learning and Image Processing. He has got 10 years of teaching experience. He has published 3 research papers in various international journals.E-mail: lame2002@gmail.com*
- *Gizachew Melkamu Molla received his B.Sc. Degree in Computer Science from University of Gondar in 2009 and his M.Sc. in Computer Science in 2015, form School of Computing, IOT, Bahir Dar University, Ethiopia. He is now a Lecturer in Debre Markos University since 2012. His main research interest is Algorithms, Cryptography, Data Mining and Cloud Computing. He is a member of professional society like SDIWC.E-mail: gizlove@gmail.com*

stack, in a queue, operations are performed at different sides of the queue. Removing elements is done at the front side and adding elements is done at the end of the queue. The element inserted lastly has to wait until all objects before it on the given queue are removed. Hence, a queue is an FIFO (First In First Out) structure.

2 Related Review

One can define tree as a connected graph with no cycles. In another words, it is a graph in which each child has only one parent except the root node. It is a set of vertices (also called nodes) and edges (or lines) in which there is only one path between any two vertices. Rooted tree has the following structure: (a) One node distinguished as root which is drawn at the top, (b) every node except the root is connected with other some node by an edge, (c) All nodes are connected with the root, and there is a unique path from the root to the each node.

In [3, 4, 12], there is a tree in which each node has at most two children (left child and right child) which is termed as binary tree. As stated there, it is different recursively as either empty or consists of a root, a left tree, and a right tree. The left and right trees in a binary tree may be empty, hence a node with one child could have a left or a right child in such tree type.

Binary tree has different applications. Some of these applications are stated in papers [3, 6, 12]. In [3], binary search algorithm is implementing for data access by the aid of binary trees. Also in [6, 12], operating system files are keeping in trees and/or tree like structures. A binary tree is applicable in compiler design for constructing syntax tree to parse expressions. In addition, it is useful in text processing, searching algorithms and evaluating mathematical expressions.

In representing mathematical expression by binary trees, we store values of the expression on leaf nodes and operators on internal nodes of the tree. There are three mechanisms for traversing this tree, namely inorder, preorder and postorder traversals. The traversing methods for these mechanisms is described papers [1, 7, 9], stated as follows.

In preorder traversal method, first visit a parent node, followed by visiting left child/subtree, and finally traversing right child/subtree. In case of inorder traversal method, we first traverse the left child, then traversing the parent node, and then traversing right child. Where as in postorder traversal, left subtree is firstly visited, next right subtree is traversed, finally visit the parent node is done.

Paper [12] presents a way to construct a binary search tree from a given traverse. It proposes an algorithm that can construct the binary search tree from only preorder traversal. This algorithm is not working for mathematical expressions, rather it considers all nodes of the tree as numeral values only. And, in a binary search tree, values of nodes in left tree and/or subtree is always less or equal than the value at the root node; and values of the right subtree is greater or equal to the root node.

Also paper [5], presents algorithm to construct a binary tree from its inorder and preorder traversals. Given these two traversals of a tree and assuming that a tree is labeled with symbols from an ordered alphabet, then it proposes algorithm to construct a binary tree. Like that of [12], it doesn't consider for mathematical expressions containing operators and operands.

[11] Presents a detail, clear step and procedure of shunting-yard algorithm [8]. Shunting-yard algorithm is used for parsing mathematical expressions given in infix notation. It is also helpful for changing the infix mathematical notations to their corresponding prefix notations.

3 Proposed Method

The different traversal methods discussed above are used for generating different notations of mathematical expressions. The preorder traversal method is used to generate mathematical expression in prefix notation. That of inorder traversal method is used to generate mathematical expression in infix notation. And, postorder traversal is used to generate mathematical expression in postfix notation. When a mathematical expression is represented using binary tree, then operands

goes to leaf nodes and that of operands are placed at internal nodes.

Under this section, we are going to present the details of our proposed method that can directly construct a binary of mathematical expressions without changing the expression to another notations like preorder or postorder notations. For our algorithm, two stacks, one for holding operands and another for holding operators are needed. The details of the algorithm is presenting below.

= == = = = = = = = = = = = = =

❖ Initialize operand and operator stacks to empty.

While there is more token in the expression, do the following.

1. Read next token from the expression if there exists.
   1.1. If it is operand
   - Push to a stack (operand stack)
   - If it is not the last operand in the expression, go back to step 1, else go to step 3.

   1.2. If it is operator, push to a stack (operator stack).
2. If operator stack is not empty and top of operator stack is not '(', do the following.
   ❖ Pop operator from operator stack if it is not ')'.
   2.1. If there is no root/internal node created with at least one leaf node, then do the following.
   - Make root node with this operator.
   - Pop operand from operand stack and make it as left child of root node.
   - Go to step 1 of this algorithm.

   2.2. If top of operator stack is '(', do the following.
      2.2.1. If its precedence is less than previously created internal node, then do the following.
      - ➢ Make it as right child of previously created internal node.
      - ➢ Pop operand from operand stack and make it left child of the new internal node.
      - ➢ Go to step 1 of this algorithm.

      2.2.2. Otherwise, go to step 2.5

   2.3. If top of operator stack is ')' and operand stack is not empty, do the following.
   - Pop operand from operand stack & make right child for newly created internal node.
   - Go to step 1 of this algorithm.

   2.4. If top of operator stack is ')' and operand stack is empty, do the following.
      2.4.1. If there exists only one internal node (root only) in the tree, do the following.
      - ➢ If its precedence is greater than that of root node, do the following.
        - o Make new root node with this operator.
        - o Make the previous root node as left child of the new root node.
        - o Go to step 1 of this algorithm.
      - ➢ Otherwise, go to step 2.5

      2.4.2. Otherwise, do the following.
      - ➢ If its precedence is less than that of root node, do the following.
        - o Make new root node with this operator.
        - o Make the previous root node as left child of the new root node.
        - o Pop matching operators '('and ')' from operator stack and delete them from the operator stack.
        - o Go to step 1 of this algorithm.
      - ➢ If its precedence is greater than parent of previously created node, do the following.
        - o Create new internal node with this operator and make it as new right child for parent of previously created internal node.
        - o Make previously created internal node as left child of this new internal node.
        - o Pop matching operators '('and ')' from operator stack and delete them from the operator stack.
        - o Go to step 1 of this algorithm.

   2.5. Otherwise, do the following.
      2.5.1. If root node has no right child and operator precedence is greater than that of operator at root node, do the following.
      - ➢ Make this operator as right child of root and also internal node of the tree.

- Pop operand from operand stack & make it as left child of this new internal node.
- Go to step 1 of this algorithm.

2.5.2. If root node has no right child and operator precedence is less or equal to the operator at root node, do the following.

- Make new root node with this operator.
- Make the previous root node as left child of the new root node.
- If operand stack is not empty, pop operand and make it as right child for the internal or root node created before this new root node.
- Go to step 1 of this algorithm.

2.5.3. If its precedence is greater than parent of previously created node, and less or equal to previously created internal node, do the following.

- Pop operand and make right child for previously created internal node.
- Create new internal node with this operator, and then do the following.
  - Make this new internal node as right child for parent of previously created node.
  - Make the previous right child of the parent together with its right child as left child of this new internal node.
  - Go to step 1 of this algorithm.

2.5.4. If its precedence is greater than both parent of previously created node and previously created internal node itself, then do the following.

- Create internal node and make it as right of the previous internal node.
- If operand stack is not empty, pop operand and make left child of the new internal node.
- Go to step 1 of this algorithm.

3. If token is the last in the expression

3.1. If operand stack is not empty

- Pop operand from operand stack
- Make right child for the latest & previous internal/root node created.

4 RESULTS AND DISCUSSION OF THE STUDY

Different operator types might exist in mathematical expression. Different operators, their descriptions, precedence and their associative is stated in paper [1, 9]. Among these operators, some of them that might work in the above are the following.

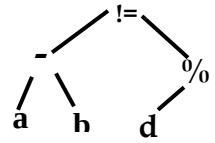
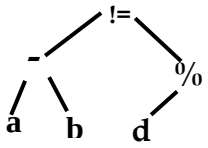
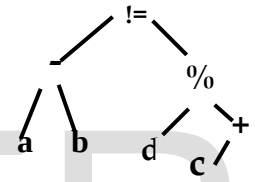
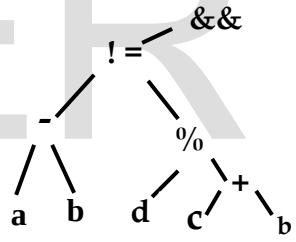
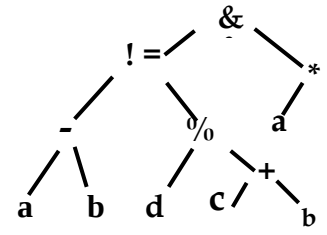
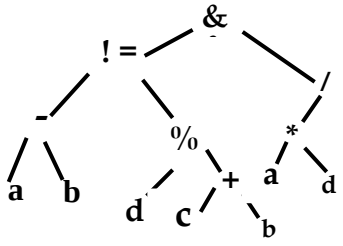
TABLE 1: OPERATORS: PRECEDENCE AND ASSOCIATIVITY

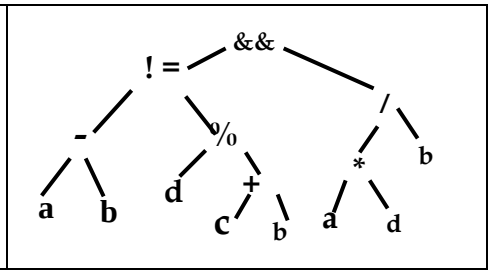
Precedence	Operator	Description	Associativity
1	* / %	Multiplication, division, and remainder	Left to right
2	+ −	Addition and subtraction	
3	<< >>	Bitwise operator : left shift and right shift	
4	< <=	Relational operators: < and ≤	
	> >=	Relational operators: > and ≥	
	= = !=	Relational operators: = and ≠	
5	&	Bitwise AND	
	^	Bitwise XOR	
	|	Bitwise OR	
6	&&	Logical AND	
	||	Logical OR	

To have a better understanding of the proposed algorithm for converting mathematical expressions to expression tree, let us have a look using example.

Given an expression: a - b != d % (c + b) && a * d / b , then find an expression tree for this expression.

TABLE 2: EXPRESSION TREE FORMATION

Expression	Algorithm steps	Operand stack	Operator stack	Expression tree
a - b != d % (c + b) && a * d / b	1.1, 1.2, 2.1	a	-	
a - b != d % (c + b) && a * d / b	1.1, 1.2, 2.5.2	b	! =	
a - b != d % (c + b) && a * d / b	1.1, 1.2, 2.5.1	d	%	
a - b != d % (c + b) && a * d / b	1.1, 1.2, 2.2.1	c	(+	
a - b != d % (c + b) && a * d / b	1.1, 1.2, 2.3, 2.4.2	b	( ) &&	
a - b != d % (c + b) && a * d / b	1.1, 1.2, 2.5.1	a	*	
a - b != d % (c + b) && a * d / b	1.1, 1.2, 2.5.3	d	/	

a - b != d % (c + b) && a * d / b	1.1	b		

5 CONCLUSION AND RECOMMENDATIONS

In this paper, we tried to show you how a given infix mathematical expression is converted to an expression tree. The algorithm developed is the first algorithm which changes a mathematical expression (infix form) to an expression tree directly with no aid of other forms like prefix and postfix forms. Once a given expression is expressed in a binary tree form, it can easily be converted to other forms of expression (prefix and post fix notations) so that the final result of the expression can easily be evaluated. Moreover, it can handle and solve the problem in evaluating operator precedence in a given mathematical expression. As this algorithm is the first attempt, it has its own limitation. The algorithm is assumed to work for binary operators which are operating on two operands. The other limitation is that, the operators assumed in the expression are all have associativity in the direction of left to right. Hence, making this algorithm to be able to work for unary & ternary operator types, and operators with associativity of right to left, is a possible future work related with this study.

REFERENCES

[1] Adam Drozdek, Data Structures and Algorithms in C++ 2nd edition (USA: Brooks/Cole, 2001).

[2] Ellis Horowitz, Sahni, Dinesh Mehta, Fundamentals of Data Structures in C++ (Galgotia, 2006).

[3] Erkki Makinen, Constructing a Binary Tree Efficiently from its Traversals, University of Tampere, Finland.

[4] E.W. Dijkstra, ALGOL Translation, Mathematics Centrum, Amsterdam, 1961.

[5] Juan Soulié, C++ Language Tutorial, cplusplus.com, 2008.

[6] Manoj C. Lohani, Upendra S. Aswal and Ramesh S. Rawat, "Reconstruction of a Binary Search Tree from its Preorder Tree Traversal with the Unique Non-recursive Approach", Oriental Journal of Computer Science & Technology, Vol. 4(1) , 217 - 219, 2011.

[7] Nishant Doshi, Tarun Sureja, Bhavesh Akbari, Hiren Savaliya, Viraj Daxini, "Width of a Binary Tree", International Journal of Computer Applications,Vol. 9, No.2, 41- 43, 2010.

[8] Nitin Arora, Pradeep Kumar Kaushik, Satendra Kumar, "Iterative Method for Recreating a Binary Tree from its Traversals", International Journal of Computer Applications, Vol. 57, No.11, 6-13, 2012.

[9] Robert L. Kruse, Data Structures and Program Design in C++, (Prentice-Hall, 1999), 321- 401.

[10] Suri Pushpa, Prasad Vinod, "Binary Search Tree Balancing Methods: A Critical Study", International Journal of Computer Science and Network Security, Vol.7, No.8, 237 – 243, 2007.

[11]The free encyclopedia, Wikipedia, available at http://en.wikipedia.org/wiki/Shunting_yard_algorithm.

[12] Vineet Kumar Sharma, Adesh Kumar Pandey, Dr. V.K. Srivastava, "A New Look to Traversal Algorithms Using Set Construct Data Structure", International Journal on Computer Science and Engineering, Vol. 02, No. 05, 1445-1448, 2010.