Equivalence between the Number of Binary Trees, Stack Permutations and Chain Matrix Multiplication

Vishal Gupta
Department of Computer Science and Engineering
G. B. Pant Engineering College
Pauri, Uttarakhand, INDIA
Email: vishalgupta87@gmail.com

Neha Bora
Department of AIM and ACT
Banasthali University
Banasthali, Rajasthan, INDIA
Email: nehabora111@gmail.com

Nitin Arora
Department of Computer Science and Engineering
G. B. Pant Engineering College
Pauri, Uttarakhand, INDIA
Email: nitinarora47@gmail.com

Abstract— **A tree is a fundamental structure in computer science. Almost all operating systems store files in trees or tree like structures. Trees are also used in compiler design, text processing and searching algorithms. A binary tree is an ordered tree in which each node has maximum of two children, referred to as left and right tree. A binary tree can either be empty or recursively consists of a root, left sub tree and right sub tree. Stack permutation refers to all the permutations that can be generated by using 3 stacks. The first stack has the numbers n, n-1... 2, 1 on it in order from bottom-to-top (i.e., the first number you can pop off the stack is 1, then 2 … then n). The second and third stacks are empty. Chain Matrix Multiplication refers to multiplication of more the two matrices and reduces the total number of multiplications. In this paper we find out the similarities between the Number of Binary Trees, Stack Permutations and Chain Matrix Multiplication.**

Keywords- Binary Tree; Chain Matrix Multiplication; Similarities; Stack Permutations.

I. INTRODUCTION

The tree is a fundamental structure in computer science. Almost all operating systems store files in trees or tree like structures. Trees are also used in compiler design, text processing and searching algorithms. A binary tree is an ordered tree in which each node has maximum of two children, referred to as left and right tree. A binary tree can either be empty or recursively consists of a root, left sub tree and right sub tree [1].

Commonly there are three traversing methods:

A. In-order Traversal

To traverse a non-empty binary tree in in-order (or symmetric order), we perform the following three operations:

1. Traverse the left sub-tree in in-order.
2. Visit the root.
3. Traverse the right sub-tree in in-order.

B. Pre-order Traversal

To traverse a nonempty binary tree in pre-order, we perform the following three operations:

1. Visit the root.
2. Traverse the left sub-tree in pre-order.
3. Traverse the right sub-tree in pre-order.

C. Post-order Traversal

To traverse a nonempty binary tree in post-order, we perform the following three operations:

1. Traverse the left sub-tree in post-order.
2. Traverse the right sub-tree in post-order

3. Visit the root.

In an in-order traversal first the left child is processed recursively, then processes the current node followed by the right child. Similarly in the preorder traversal first the root is processed followed by the left child and the right child [2][3].

Figure 1 illustrates binary tree and its traversals in pre-order, in-order, and post-order. [4]

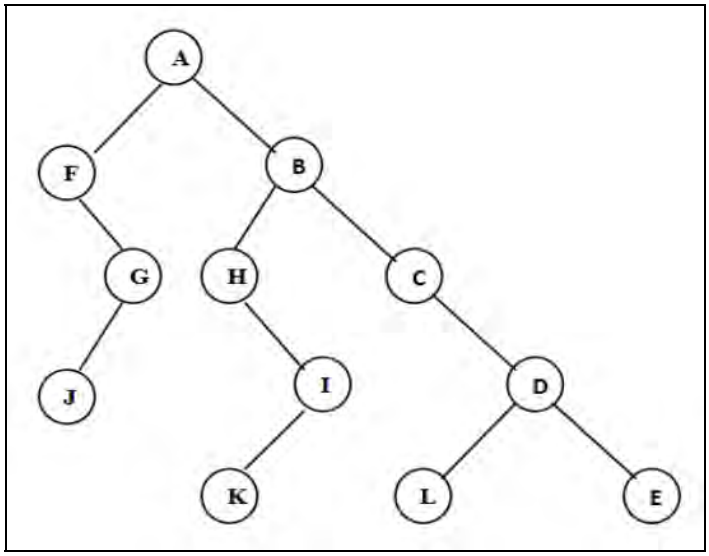

Figure 1: Binary Tree containing 12 nodes

Pre-order traversal: A, F, G, J, B, H, I, K, C, D, L, E
In-order traversal: F, J, G, A, H, K, I, B, C, L, D, E
Post-order traversal: J, G, F, K, I, H, L, E, D, C, B, A

Stack permutation refers to all the permutations that can be generated by using 3 stacks. The first stack has the numbers n, n-1.... 2, 1 on it in order from bottom-to-top (i.e., the first number you can pop off the stack is 1, then 2 … then n). The second and third stacks are empty. We are allowed two kinds of moves. We can either pop a number off the first stack and push it onto the second stack, or pop a number off the second stack and push it onto the third stack. "Backwards" moves are not allowed [5].

Another problem that has a connection with the previous two involves the product of two matrices [6]. Suppose we wish to compute the product of n matrices M_1, M_2,…,M_n

Since matrix multiplication is associative, we can perform these multiplications in any order. We would like to know how many different ways we can perform these multiplications [15]. For example, if n=3, there are two possibilities (M_1 * M_2) * M_3 and M_1* (M_2 * M_3)

II. Equivalence between the Number of Binary Trees, Stack Permutations and Chain Matrix Multiplication

Stack permutation refers to all the permutations that can be generated by using 3 stacks. The first stack has the numbers n, n-1... 2, 1 on it in order from bottom-to-top (i.e., the first number you can pop off the stack is 1, then 2 … then n). The second and third stacks are empty. We are allowed two kinds of moves [12]. We can either pop a number off the first stack and push it onto the second stack, or pop a number off the second stack and push it onto the third stack. "Backwards" moves are not allowed [7][9][10]. We can move all the numbers from the first stack to the third. All the sequence of numbers that can be generated starting from first stack to the third stack on the third stack is known as a stack permutation. For example,

Permutations of n=3 numbers on 3 stacks...

Moves: 1->2; 1->2; 1->2; 2->3; 2->3; 2->3;
Finished value: 1, 2, 3

Moves: 1->2; 1->2; 2->3; 1->2; 2->3; 2->3;
Finished value: 1, 3, 2

Moves: 1->2; 1->2; 2->3; 2->3; 1->2; 2->3;
Finished value: 3, 1, 2

Moves: 1->2; 2->3; 1->2; 1->2; 2->3; 2->3;
Finished value: 2, 3, 1

Moves: 1->2; 2->3; 1->2; 2->3; 1->2; 2->3;
Finished value: 3, 2, 1

So the permutations for n=3 are (3,2,1), (2,3,1), (2,1,3), (1,3,2), (1,2,3)

The permutation values are read from bottom to top. So as we can see the value (3, 1, 2) cannot be generated for n=3 numbers.

Another problem that has a connection with the previous two involves the product of two matrices. Suppose we wish to compute the product of n matrices M_1, M_2,…,M_n

Since matrix multiplication is associative, we can perform these multiplications in any order. We would like to know how many different ways we can perform these multiplications. For example, if n=3, there are two possibilities (M_1*M_2) * M_3 and M_1* (M_2 * M_3)

Let b_n be the number of different ways to compute the product of n matrices. Then b_2=1, b_3=2 and b_4=5. Let M_{ij}, i<=j me the product of M_i * M_{i+1} * … * M_j. The product we wish to compute is M_{1n}.We may compute M_{1n} by computing any of the products M_{1i} * $M_{(i+1)n}$, 1<=i<=n and this can be done recursively. The number of distinct ways to obtain M_{1i} and $M_{(i+1)n}$ are b_i and $b_{(n-1)}$ respectively[8]. Therefore letting b=1, we have

$$b_n = \sum_{i=1}^{n-1} b_i * b_{i-1}, n > 1 \quad (1)$$

If we can determine the expression for b_n only in terms of n then we have a solution to our problem. Also, If the nodes of a tree are numbered such that its preorder permutation is 1, 2,…, n then the number of distinct binary trees having the preorder permutations 1, 2,…, n. Using the concept of an in-order permutation, we can show that the number of distinct permutations obtainable by passing the numbers 1 to n through a stack and deleting in all possible ways is equal to the number of distinct binary trees with n nodes [2][11].

If we start with the numbers 1, 2, 3 then the possible permutations obtainable by a stack are:

(1,2,3) (1,3,2) (2,1,3)(2,3,1)(3,2,1)

Obtaining (3, 1, 2) is impossible.
Figure 2 shows the each of these 5 permutations corresponds to one of the five distinct binary trees with three nodes.

Now instead of b_n be the number of distinct binary trees with n nodes. Again an expression for b_n in terms of n is what we want [10]. Then we see that b_n is the sum of all the possible binary trees formed in the following way: a root and two sub trees with bi and $b_{(n-i-1)}$ nodes, for 0<= i < n . This explanation says that

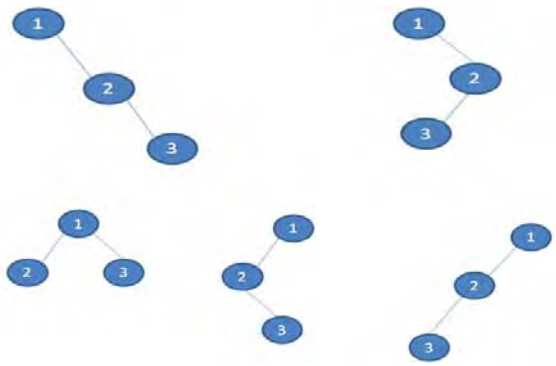

Figure 2: Binary Trees corresponding to five permutations.

$$b_n = \sum_{i=1}^{n-1} b_i * b_{n-i-1}, n > 1, and\ b_0 = 0 \quad (2)$$

This formula and the previous one are essentially the same. Therefore, the number of binary trees with n nodes, the number of permutations of 1 to n obtainable with a stack, and the number of ways to multiply n+ 1 matrix are all equal [13][14].

The formula for calculating the number of stack permutations having n values is

$$b_n = 1/(n+1) * C(2n, n) \quad (3)$$

This is approximately of the order $b_n = O(4^n / n^{1.5})$

III. CONCLUSION

From the above results 1, 2 and 3, it can be concluded that the number of binary trees with n nodes, the number of permutations of 1 to n obtainable with a stack, and the number of ways to multiply (n+1) matrices are all equal.

REFERENCES

[1] Knott. G.D. “A numbering system for binary Trees”, Comm ACM 20 (1977)113-115.
[2] Hu, T C.; M T Shing (1984). “Computation of matrix chain product”. Part II”. SIAM Journal of Computing (Univ. of California at San Diego: Springer-Verlag) 13 (2): 228-251, doi 10.1137/0213017. ISSN 0097-5397.
[3] Suresh Kumar, Vishal Gupta and Vivek Kumar Tamta; “Dynamic Instruction Scheduling for Microprocessors Having Out Of Order Execution” Computer Engineering and Intelligent Systems, IISTE, pp. 10–14, Vol. 3, No. 4, 2012.
[4] Nitin Arora, Vivek Kumar Tamta, Suresh Kumar: “Modified Non-Recursive Algorithm for Reconstructing a Binary Tree”, International Journal of Computer Applications (0975 – 8887) Volume 43– No.10, April 2012.
[5] Suresh Kumar, Madhu Rawat, Vishal Gupta and Satendra Kumar; “The Novel Lossless Text Compression Technique Using Ambigram Logic and Huffman Coding” Information and Knowledge Management, IISTE, pp. 25-31, Vol. 2, No. 2, 2012.
[6] Nandita Goyal Bhatnagar, Vishal Gupta and Anubha Chauhan; “A Comparative Study of Differentiation Between Macintosh and Windows Operating System” IJREISS, pp. 77-85, Vol. 2, Issue 6, June 2012.
[7] Nitin Arora, Vivek Kumar Tamta, Suresh Kumar: “A Novel Sorting Algorithm and Comparison with Bubble sort and Insertion sort”, International Journal of Computer Applications (0975 – 8887) Volume 45– No.1, May 2012.
[8] Robert Sedgewick: “Algorithms in C”, Addison Wesley, December 1990.
[9] Seymour Lipschutz: “Theory and problem of Data Structures”, International Edition, McGRAW-HILL, 1986.
[10] Sartaj Sahni: “Data Structures, Algorithms and Applications in JAVA”, 2nd Ed., University Press.
[11] Mark Allen Weiss, “Data Structures and Algorithm Analysis in C”, Vol. 3 (2nd ed.), Addison-Wesley, 1997.
[12] R. Sedgewick, Algorithms in Java, 3d edition, Addison Wesley, 2003
[13] C. C. Lee, D. T. Lee, C. K. Wong, Generating Binary Trees of bounded height, Acta Inf., 23, 529-544, 1986.
[14] Michael Main, Walter Saitch: “Data Structures and Other Objects” Turbo Pascal Edition, ISBN: 0-8053-7086-2.
[15] Seymour Lipschutz: “Theory and problem of Data Structures”, International Edition, McGRAW-HILL, 1986.