International Journal of Scientific Research and Management (IJSRM)
||Volume||5||Issue||07||Pages||5953-5958||2017||
Website: www.ijsrm.in ISSN (e): 2321-3418
Index Copernicus value (2015): 57.47 DOI: 10.18535/ijsrm/v5i7.22

An Inquistive Result on DFS Problem of Binary Trees

[1]Dr. N. Rama, [2]Justin Sophia. I

[1]Associate Professor, Post Graduate and Research Department of Computer Science,
Research scholar , Post Graduate and Research Department of Computer Science,
Presidency College, Chennai 600005

Abstract:Let T be a complete binary tree of height h,then T is called a *full binary tree*, if all the leaves in T are at level h. Let FBT_n denote the full binary tree of height n.Let $DFS(x,G)$ denote the time taken by DFS with respect topreorder traversing to find the vertex x in the graph G. In this paper we analytically compute the $DFS(x,G)$ for G being a full binary tree of height n, FBT_n.

Keywords: Binary Tree, DFS, Time complexity.

1. INTRODUCTION

Generally this paper deals with elements of interconnected network through a data structure with a mathematical formulation.[1] In the field of computer science, a data structure and algorithm plays a vital role in bringing mathematical formulation in designing a effective and efficient software. For the classification of nodes we use tree structures where Trees can hold objects that are sorted by their keys[2]

DFS: DFS refers as depth first search; it is another technique to traverse the graph. This algorithm starts from top to bottom. To implement this algorithm stack is used. Basic steps of algorithm are: 1) Push the bottom element of the graph in the stack. 2) Pop the top element fromthe stack. 3) Push non visited node in the stack and repeat steps again until stack is empty.

Depth First Search (DFS) is a tree or graph traversal algorithm optimized for faster searching [4]. The searchtypically commences at the root of the tree and continues to explore each branch till the leaf nodes, before backtracking. As soon as the search reaches the leaf node/dead end, the control returns to the most recent node that has not been explored. DFS is closely related to preorder traversal of atree. A preorder traversal simply visits each node before its tree. A preorder traversal simply visits each node before its graph traversal algorithm, the "child" node is replaced by "neighbor" node. Further, in order to optimize the algorithm and prevent infinite loops, one would restrict the number of visits to each vertex. In order to enforce this restriction, marks are used to keep track of the visited vertices. This search may be used to build a spanning tree with certain useful properties.

The Depth First Search (DFS) algorithm begins with initializing a set of markers in order to identify the pre-visited vertices. Upon choosing a starting vertex 'x', a tree Tis initialized to x, and Depth First Search(x) algorithm is invoked. The order of traversal of the vertices beginning from a vertex having multiple neighbors makes littledifference. The graph shown in Figure 1 illustrates theapplication of the DFS algorithm.

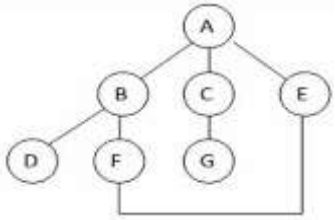

Figure. 1. Graph Used for Applying DFS

Assuming the left edges in the graph are picked prior to the right edges while performing a DFS, commencing atvertex 'A', for a search that presumably memorizes pre- visited nodes (in order

DOI: 10.18535/ijsrm/v5i7.22

not to repeat them), the nodes will be ordered as A, B, D, F, E, C and G. The result of this search is a structure known as the Trémaux tree. Performing the same search without any knowledge of pre-visited nodes accounts for a different order of nodes : A, B, D, F, E, A, B, D, F, E etc. with infinite loops caught in the A, B, D, F, E cycle and inability to reach nodes C or G.

During tree traversal, for space complexity evaluation, the memory needed in DFS is the largest when the searchreaches the maximum tree depth for the first time.

Assuming, a branching factor 'b' for each node, when a node at depth 'd' is inspected, the total number of nodes/vertices collected in memory is a combination of unexpanded nodes up to depth 'd' and the current node being studied. With b-1 number of unexpanded nodes at each level, the total amount of memory that is required iscomputed as d *(b -1) +1 i.e. O(|V|) , if the entire graph is traversed without repetition. To summarize, the spacecomplexity of Depth First Search is a linear function of thebranching factor for each node and this makes DFS morefavorable over Breadth First Search that has an exponentialfunction. For evaluating time complexity, the paper takesinto consideration the time needed in searching a node. In order to search a node located at the leftmost location at a

depth d in a tree results in the total number of examinednodes to just being just d +1. On the other hand, if one findsthe node at the extreme right location at depth d, then thetotal number of inspected nodes comprises of all the nodesin the tree and it is calculated as: 1 + b + b + b +….+ b =2 in the tree and it is calculated as: 1 + b + b + b +….+ b =2 in the tree and it is calculated as: 1 + b + b + b +….+ b =in the tree and it is calculated as: 1 + b + b + b +….+ b =d.(b^{d+1} – 1) / (b – 1). In an average case scenario, the total number of examined nodes is evaluated as [(d + 1) / 2] +[(b^{d+1}– 1) / (b – 1)] ≈ [b (b + d)/2(b – 1)]. Asymptoticallyspeaking, the run-time complexity is evaluated as O(b^{d})branching factor b and depth d.

2. GRAPH PRELIMINARIES

Graph theory is a branch of mathematics initiated with the Konisberg bridge problem, in the 18th century which was solved by the famous Swiss mathematician Leonhard Euler. Graphs are very powerful tools for representing the relationships among objects. The objects are represented by vertices and the relationship among the vertices is represented by edges. Graphs contribute themselves naturally as models for various situations. For example, consider the set of towns and railline system connecting the towns. Here, the points represent towns and the lines represent direct trains between certain pairs of these towns. Multiprocessor computers or switching circuits, electrical networks can also be represented by graphs.

An interconnection network is important for fast and reliable communication among the processing nodes in any parallel computer. Graph theory is a valuable tool for the design and analysis of the structural properties of interconnection networks. Graph theory has rigorous applications in computational biochemistry, chemistry, operations research, game theory, computer science, genetics, chemical, electrical, civil and mechanical engineering. Algebraic graph theory is an extension of graph theory in which algebraic methods are applied to problems of graphs. Several books have also been published on applied graph theory.

For graph theoretic definitions and notations, we refer to [].

A pair $G = (V, E)$ with $E \subseteq (V \times V)$ is called a graph (on V). The elements of V are the vertices of G, and those of E the edges of G. The vertex set of a graph G is denoted by $V(G)$ and its edge set by $E(G)$. In literature, graphs are also called simple graphs; vertices are called nodes or points; edges are called lines or links. The list of alternatives is long (but still finite). A pair $\{u, v\}$ is usually written simply as uv. Notice that then $uv = vu$. In order to simplify notations, we also write $v \in G$ and $e \in G$ instead of $v \in V(G)$ and $e \in E(G)$.

A graph H is a subgraph of a graph G, denoted by $H \subseteq G$, if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. A subgraph $H \subseteq G$ spans G (and H is a spanning subgraph of G), if every vertex of $G$is in H, i.e., $V(H) = V(G)$. Also, a subgraph $H \subseteq G$ is an induced subgraph, if $E(H) = E(G) \cap E(V(H))$. In this case, H is induced by its set $V(H)$ of vertices.

DOI: 10.18535/ijsrm/v5i7.22

An $(u; v)$-path is called to be *shortest* if its length is the smallest over all $(u; v)$-paths in G. The distance from u to v, denoted by $d(G; u; v)$ is defined as the length of the shortest $(u; v)$-path in G. The *diameter* of G, denoted by $d(G)$, is de ned as the maximum distance from any one vertex of G to any other. A *planargraph* is a graph which can be drawn on the plane so that its edges cross only at theirend-vertices.

A graph is acyclic, if it has no cycles. A tree is a connected acyclic graph. If deg v = 1 then v is called a leaf. All other vertices are internal vertices. Any two vertices of a tree are connected by a unique path. A tree in which one vertex, the root, is distinguished, is called a rooted-tree and the remaining vertices are partitioned intodisjoint sets $T_1, T_2, \ldots, T_n$, where each of these sets is a tree. The subtrees ofthe root are the sets $T_1, T_2, \ldots, T_n$. The roots of the subtrees of a vertex x are thechildren of x. The vertex x is referred as the parent of its children. Children of the same parent are called siblings. k-ary tree is the most common type of tree in which each node can have at most k descendants or children. A k-ary tree is said to be a complete if each internal vertex has exactly k descendants.

3. FULL BINARY TREE

A rooted tree T is called an m-*ary tree* if every vertex of T has at most m children. A rooted tree T is called a *complete* m-*ary tree* if every internal vertex of T has at most m children. A *complete binary tree* is a 2-ary tree in which all leaves have the same depth and all internal vertices have degree 3, except the root. If T is a rooted tree and h is the largest level number achieved by a leaf of T, then T is said to have height h. A rooted tree of height h is said to be *balanced* if the level number of every leaf is h or $h-1$. Let T be a complete binary tree of height h,then T is called a *full binary tree*, if all the leaves in T are at level h. Let FBT_n denote the full binary tree of height n.

In FBT_n, the root of each subtree is said to be a "*child*" of r, and r is the "*parent*" of each subtree root. Each node has exactly one parent; the *root* has no parent. *Siblings* are the pair of nodes with the same parent. *Ancestor* are any node on the path between the root and the specified node, including the root. *Descendent* are any node in the subtree that has the specified node as its root. *Leaves* are nodes with no children. *Internal node* are nodes with two children.

In the paper, we label $V(FBT_n)$ by natural numbers level wise as shown in fig. 1.1

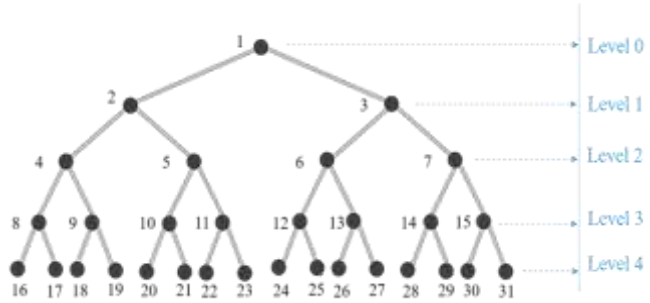

Figure.2. Full Binary Tree (FBT_4)

4. DEPTH / BREATH FIRST SEARCH

Traversing Binary Trees can be done by *depth first search DFS* or by *breath first search BFS*. The DFS are of three types which rely on a stack. The prefix of the names of the traversals indicates when the parent is processed. *Preorder*: parent, then left child, and then right child. *Inorder*: left child, then parent, and then right child. *Postorder*: left child, then right child, and then parent. Level order traversal or the BFS relies on a queue: parent, children of parent, grandchildren of parent and so on.

In Depth First Search, go as far as possible along a single path until reach a dead end (a vertex with no edge out or no neighbor unexplored) then backtrack. The application of depth first search is topological sort.DFS follows the following rules: Select an unvisited node x, visit it, and treat as the current node. Find an unvisited neighbor of the current node, visit it, and make it the new current node; If the current node has no unvisited neighbors, backtrack to the its parent, and make that parent the new current node; Repeat previous two steps until no more nodes can be visited. If there are still unvisited nodes, repeat initial step.

Breadth-first Search (BFS) is a fundamental graph theory algorithm that is extensively used to abstract various challenging computational

DOI: 10.18535/ijsrm/v5i7.22

problems. Due to the fine-grained irregular memory accesses, parallelization of BFS can exhibit limited performance on cache-based systems.

Breadth first search is a general method used to search the path with minimum number of edges between two given vertices. BFS is a traversal graph G which visits all vertices and edges in a graph. The breadth first search does the search level by level.

Breadth-first Search is a graph traversal algorithm on an undirected graph. Let G = (V, E) denote the input undirected graph, where V = {0, 1, · · · , N − 1} is the node set and E is the edge set. We assume that G is connected. Given G and a root node s ∈ V , BFS begins at s and ex- plores all the neighbor nodes. Then, for each of the neighbor nodes, BFS visits the unexplored neighbor nodes, and so on, until it traverses all the nodes. During this procedure, all the nodes at the same distance to s, i.e., the nodes at the same level, must be explored before their neighbor nodes in the next level can be explored

5. DFS TIME COMPLEXITY IN FBT_n

Let $DFS(x,G)$ denote the time taken by DFS by preorder traversing to find the vertex x in the graph G. In this section we discuss the $DFS(x,G)$ for G being a full binary tree of height n, FBT_n.

Theorem 1: Let $x \in V(FBT_n)$ and let $x = x_1, x_2, \ldots, x_k = 1$ be the sequence such that $x_{i+1} = \left\lfloor \frac{x_i}{2} \right\rfloor, 1 \le i < k$. Then,

$$DFS(x,FBT_n) = 2^{n+2}\left(1 - \frac{1}{2^{\lfloor \log_2 x \rfloor}}\right) - k + 1 - \sum_{i=1}^{k}\left((1-(-1)^{x_i-1}) \times \left(2^{n+1-\lfloor \log_2 x_i \rfloor} - 1\right)\right)$$

Proof. Given that $x \in V(FBT_n)$. Clearly the sequence $x = x_1, x_2, \ldots, x_k = 1$ where $x_{i+1} = \left\lfloor \frac{x_i}{2} \right\rfloor$, $1 \le i < k$ represents the shortest path from x to the root vertex 1. FBT_n has n number of levels and each l^{th} level comprises 2^l number of vertices. Thus x belongs to $\lfloor \log_2 x \rfloor^{th}$ level. FBT_n has $2^{n+1} - 1$ vertices. For x being a vertex in level l, then the number of descendent vertices of x would be

$$\left(2^{(n-l)+1} - 1\right) - 1$$

i.e.

$$2^{n+1-\lfloor \log_2 x \rfloor} - 2$$

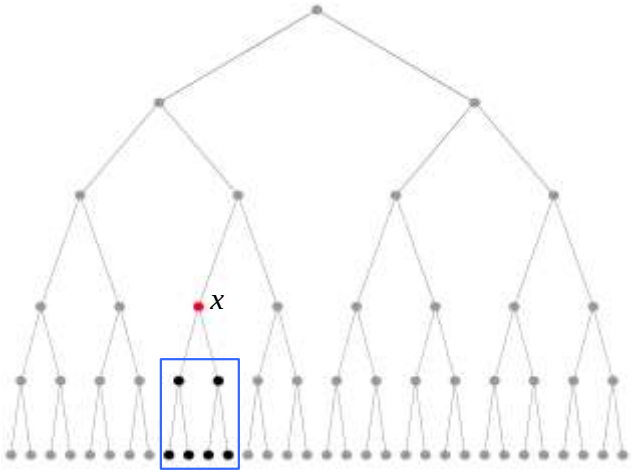

Figure.3. Descendent of vertex x

Clearly, the descendent vertices of x remains unvisited after the search of x (See Figure.3). For any even vertex say x_j in the sequence $x_1, x_2, \ldots, x_k$, its corresponding right sibling $x_j + 1$ remains unvisited (See Figure.3). More over all the descendent vertices of $x_j + 1$ remains unvisited. To sum up,

$$\frac{1}{2}\sum_{i=1}^{k}\left((1-(-1)^{x_i-1}) \times \left(2^{n+1-\lfloor \log_2 x_i \rfloor} - 1\right)\right)$$

remains unvisited.

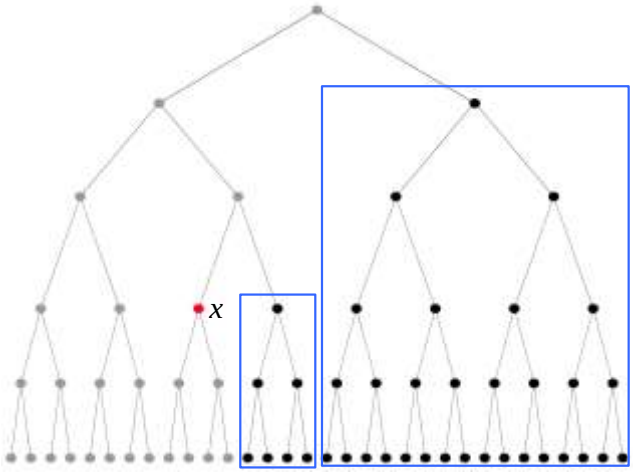

Figure.4. Right sibling along with their descendent for every even vertices in the sequence from x to root 1.

DOI: 10.18535/ijsrm/v5i7.22

Thus the total number of vertices not visited being

$$2^{n+1-\lfloor \log_2 x \rfloor} - 2 + \frac{1}{2}\sum_{i=1}^{k}\left(\begin{matrix}(1-(-1)^{x_i-1})\times \\ (2^{n+1-\lfloor \log_2 x_i \rfloor}-1)\end{matrix}\right)$$

In other words, the total number of vertices that are visited would be

$$(2^{n+1}-1) - 2^{n+1-\lfloor \log_2 x \rfloor} - 2 + \frac{1}{2}\sum_{i=1}^{k}\left(\begin{matrix}(1-(-1)^{x_i-1})\times \\ (2^{n+1-\lfloor \log_2 x_i \rfloor}-1)\end{matrix}\right)$$

i.e.

$$2^{n+1} - 2^{n+1-\lfloor \log_2 x \rfloor} + 1 - \frac{1}{2}\sum_{i=1}^{k}\left(\begin{matrix}(1-(-1)^{x_i-1})\times \\ (2^{n+1-\lfloor \log_2 x_i \rfloor}-1)\end{matrix}\right)$$

i.e.

$$2^{n+1}\left(1-\frac{1}{2^{\lfloor \log_2 x \rfloor}}\right) + 1 - \frac{1}{2}\sum_{i=1}^{k}\left(\begin{matrix}(1-(-1)^{x_i-1})\times \\ (2^{n+1-\lfloor \log_2 x_i \rfloor}-1)\end{matrix}\right)$$

Let this total number of visited vertices be φ. The tree induced by the visited φ vertices has $\varphi - 1$ edges. In this induced tree, exactly the edges on the shortest path between x and the root vertex 1 are traversed exactly once and the remaining edges are traversed twice. Therefore, the time taken to search x by DFS is

$$2(\varphi - 1) - (k-1)$$

i.e.

$$2\varphi - k - 1$$

Replacing the value for φ we get,

$$2\left(2^{n+1}\left(1-\frac{1}{2^{\lfloor \log_2 x \rfloor}}\right) + 1 - \frac{1}{2}\sum_{i=1}^{k}\left(\begin{matrix}(1-(-1)^{x_i-1})\times \\ (2^{n+1-\lfloor \log_2 x_i \rfloor}-1)\end{matrix}\right)\right) - k - 1$$

i.e.

$$2^{n+2}\left(1-\frac{1}{2^{\lfloor \log_2 x \rfloor}}\right) - \sum_{i=1}^{k}\left(\begin{matrix}(1-(-1)^{x_i-1})\times \\ (2^{n+1-\lfloor \log_2 x_i \rfloor}-1)\end{matrix}\right) - k + 1$$

Hence the theorem.

Example: Consider BT_5 , see figure.5.

$$DFS(17, FBT_5) = 10$$
$$DFS(10, FBT_5) = 33$$
$$DFS(6, FBT_5) = 64$$
$$DFS(28, FBT_5) = 96$$
$$DFS(60, FBT_5) = 111$$

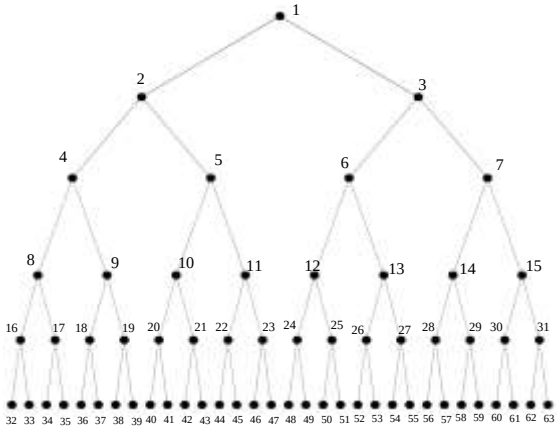

Figure.5. FBT_5

6. CONCLUSION

In this paper we discuss the time taken to search a vertex by depth first search by pre-order for a full binary tree of height n. Also, the time complexity for inorder and post order for full binary tree are under investigation. Similar computation can be extended to complete binary tree, binary tree, trinary tree and other topological networks such as grid, hypercubes and other variants for trees.

REFERENCES

[1] T . H . Cormen, C. E . Leiserson, R. L. Rivest and C. Stein Introduction to Algorithms, Cambridge, MA: MIT Press , September 2009

[2] N. Gelfand , M . T. Goodrich and R. Tammasia,"teaching Da t a Structure Design Patterns," Proceedings of the 29 SIGCSE Technical Symposium on Computer Science Education, vol. 30 (pp. 331-335) , Providence, RI, USA, 1998 .

[3] R.Bayer, "Binary B-Trees for Virtual Memory ," Proceedings of ACM SIGFIDET Workshop on Data Description, Access and Control , pp. 219-235, San Diego , CA, USA, 1971

[4] R. Tarjan, "Depth First Search and Linear Graph

Algorithms," Proceedings of the 12th Annual IEEE Symposium on Switching and Automata Theory, pp. 141-121, East Lansing, MI, USA, 1971.

[5] Everitt, Tom, and Marcus Hutter. "Analytical Results on the BFS vs. DFS Algorithm Selection Problem. Part I: Tree Search." Australasian Joint Conference on Artificial Intelligence. Springer International Publishing, 2015.

[6] Everitt, Tom, and Marcus Hutter. "Analytical Results on the BFS vs. DFS Algorithm Selection Problem: Part II: Graph Search." Australasian Joint Conference on Artificial Intelligence. Springer International Publishing, 2015.

[7] J.A.Bondy, U.S.R. Murty, Graph Theory with Applications, North Holland, New York.Amsterdam.Oxford, 1976.

[8] F. Harray. Graph Thoery. Addison. Wesley, Reading. Mass (1969).

[9] Deo, Narsingh. Graph theory with applications to engineering and computer science. Courier Dover Publications, 2016.

[10] A.Bagchi, S.Hakimi, Data Transfers in Broadcast Networks, IEEE Transactions
Comput. 41, (1992).

[11] R.Balakrishnan, K.Ranganathan, A Text Book of Graph Theory, Springer,(1999)

[12] C.Berge, The Theory of Graphs, Methuen, (1962).
[13] N.L.Biggs, E.K.Loyd, R.J.Wilson, Graph Theory, Oxford University Press,(1976), 1736 { 1936 .

[14] R.Diestel, Graph Theory, Springer, (1991).

[15] J.L.Gross, J.Yellen, Graph Theory and its Applications, CRC Press, (1998).