An Node Search of DFS with Spanning Tree in Undirected Graphs

D. Jasmine Priskilla, K. Arulanandam

Abstract: ***In a graph, spanning tree is a subgraph it is also a tree which relates all the vertices together. So it 'Spans' the first graph yet utilizing less edges Graph Search is a calculated plot that visits vertices or edges in a graph, in a request dependent on the availability of the graph. In graph search, edges are visited all things considered once and not all edges are visited. The ones that are visited structure a spanning tree for the vertices that are associated with the beginning vertex by a path. A spanning tree for a lot of vertices VER is a lot of edges without cycles that associates VER. So in a spanning tree, there is actually one path between any two of the vertices. This is the fundamental explanation behind the utility of DFS. DFS utilize stacks and creates insignificant spanning tree fulfilling an assortment of conditions.***

Keywords - shortest path, spanning tree, undirected graph, node searching, depth first search, breadth first search.

I. INTRODUCTION

An undirected graph G is denoted as (VER,EDG) where VER represented as the arrangement of m vertices and EDG represented as the arrangement of n edges. ST is a spanning tree of graph G and also related subgraph of given graph G that is likewise a tree and holds all the vertices of G. With the end goal that, a chart G has many spanning trees as a rule and it might shift depending on the structure of the graph. For instance if the given graph G is in the form of tree i.e without any cycles, at that point the graph has just one spanning tree. Be that as it may, in the event that it has cycles in the structure, at that point it might have more than one spanning tree.

A few graphs have exceptionally huge number of spanning trees, and it is characterized as past polynomial. In the event that G is associated, and the essential graph traversal techniques like Breadth-first search and Depth-first search can also give a spanning tree in a straight time. On the off chance that the separation between two vertices determined as a unit esteem, at that point in the event that it is known as weight, each spanning tree has its own combined weight value.

We can locate the minimum spanning tree an incentive from the effective calculations of Kruskal [2] and Prim [3]. These two strategies for minimum spanning tree issues are understood by in polynomial time. Be that as it may, a few calculations settled by NP-complete technique depends on the ideas accessible in [4].

A. Previous works on node searching Spanning tree problems

Previous work center around the guess approach of the idea to utilize the hub for the basic concept of developing a spanning tree and propose an estimation algorithm of O(log n) [5]. It delivered minimum spanning tree with minimum number of searches with the mix of node search an essential variation of graph search and spanning tree.

The algorithm BFSTree delivered a spanning tree T established at the inside cent and it's clear the branch recursively by regarding the hub as cent. Nopandon Juneam et al. [5] is the main composed concept that ties together graph looking and spanning tree. Nopandon Juneam et al.[5] recommend the extent of research to extend further with different variations of graph searching.

Different appraisals assessing the show of Breadth-First Search and Depth-First Search. The assessments have assumed that both of the calculations have their places of focal points and hindrances. Which inferred that there is no sensible technique to make sense of which calculation is better before evaluating the concepts in unequivocal circumstances [7,8].

B. Our results

In this paper, following the methodology of [5], we build up the new algorithm with DFSTree by consolidating the spanning tree ideas and graph search. Alexander Olsson and Therese Magnusson of [6], presumed that utilizing Breadth-First search ideas are helpful for little graphs and furthermore it expends more memory contrast with Depth-First search. Likewise they demonstrated that if Graph size is biggest Breadth-First search ought not have the option to apply, rather every rationale will be assessed case by case. From this outcomes DFS ideas are greatly improved when we search a node in undirected graphs.

C. Organization of the paper

In segment 3 we present the hypothetical foundation of the algorithm with some graph terminologies. In segment 4, we secured foundation for the picked benchmarks and graph database programming. In fragment 4.2 an appraisal of various unmistakable BFS and DFS estimations in data stream assessment by J. Cobleigh, L. Clarke and L. Osterweil is presented. This article was picked since it gives the multifaceted design of surveying search calculations and that there is no calculation that is the best for each search issue. Improving hunt calculations should in like manner be conceivable by running them all the while and this is brought into fragments 4.5.1 and 4.5.2. These two articles were used as inspiration for the headways we executed for our Breadth First Search and Depth First Search operand.

Revised Manuscript Received on January 5, 2020

D. Jasmine Priskilla, Ph.D Research Scholar, Research and Development Centre, Bharathiar University, Coimbatore, Tamilnadu, India.

Dr. K. Arulanandam, Head, Department of Computer Applications, GTM College, Gudiyattam, Tamilnadu, India.

II. PRELIMINARIES AND DEFINITIONS

In this segment we give the definition and documentations which we will use all through this paper. Wherever we examine as a graph the wording G is indicated as in-directed graph comprises of (VER,EDG) with |VER| = m vertices and |EDG| = n edges. Now let us assign the vertices of G is meant by VER(G) and edges indicated by EDG(G).

Let G(ST) is signified by a subgraph and ST is has a place with the subset of {VER} in G. The Search procedure of a graph G is meant as SS(G), and the cost of SS is referenced as cost(SS). The cost is determined from the most extreme number of searchers in G. The Graph Search number is indicated as SN(G) which is the base expense of the considerable number of searches accessible in the graph G.

Recuperating information from a gien graph is known as a traversal and incorporates "strolling" along the parts of the diagram. This is every now and again limited in qualification to SQL inquiries [11]. A credited, stamped, coordinated multi-diagram is known as a property chart and many various charts are a subset of property charts. It in like manner infers that for all intents and purposes anything may be addressed as given graph[11]. To obtain the database oriented graph result we have selected Neo4j, a open source software and worked with it.

Figure is an inquiry language for the Neo4j chart database, similar to the SQL used for social databases. A Cipher question contains different stipulations that build the inquiry. A segment of these are MATCH, WHERE, RETURN and CREATE which do exactly as expected by the names of the operations: discovering ways that match the given model, giving filtering criteria to the organized results, reestablishes the predefined qualities or makes another hub or relationship[10].

III. THEORETICAL BACKGROUND

A. Spanning Tree for Node Searching

The given graph G=(VER,EDG) produces a spanning tree and it is a subgraph TR=(VER,EDG) which is connected and also a tree with no cycle. The given graph G has many spanning trees like TR. In any case, this node looking through ideas considers the minimum cost of spanning from the arrangement of trees TR.

In graph searching area, we plan to propose an O(log n)-estimation algorithm for the node looking spanning tree issue. Our guess approach depends on the idea of utilizing hub point as a base for constructing a spanning tree. For this section's purpose, our proposed estimation algorithm, we inspected a graph G = (VER,EDG) altogether. For any pair of vertices s, t∈VER, we figure the separation dist(s, t) in $O(n^3)$ time utilizing the outstanding Floyd-Warshall algorithm. For every node s∈VER, we process the erraticism of s, e(s) = max ,t∈VER dist(s, t), in $O(n^2)$ time. We assign the graph's radius rad(G) = min, s∈VER e(s) and the graph's focuses cent(G) = {s ∈ VER | e(s) = rad(G)}, all in O(n) time.

Let T* indicates that, a spanning tree to such an extent that there exists a quest procedure SS for T* with cost(S) = s(T*). Give us a chance to accept that T* has a center point. Our estimation algorithm surmises an appropriate huband stays as hub. The suppositions depend on the thought of hub point that point should detach the tree and form as an assortment of smaller sub-trees which is cleared by utilizing the less number of searchers. In the first place, we will attempt to discover a node that patterns to carry on along these lines. To make a decent theory, we will choose a greatest degree node from the graph's focuses as our hub. We developed a spanning tree by utilizing a Breadth First Search tree established at this point. Our new approximation concept of the algorithm is provided in the accompanying.

B. Algorithm for Spanning Treed

Input: A given graph G.

Output: A new spanning tree ST of given graph G.

Procedure:

Step 1: Findout the centers of the graph as cent(G).

Step 2: c = max s ∈cent(G) deg(s).

Step 3: ST = DFSTree(G, t).

IV. ANALYSING THE RESULTS OF BFS VS. DFS ALGRITHMS WITH THE SELECTION PROBLEM

Evaluating Breadth First Search and Depth First Search are moreover intriguing inside the field of Artificial Intelligence, because of the broad extent of issues in the field of search it can arranged as looking issues. T.Everittand M. Hutter completed an examination on meta-heuristics [9], that are general request methods and which are not going for a specific sort of issue anyway rather use a neighborhood association on the game plan space similarly as a heuristic limit. For the meta-heuristics both Breadth First Search and Depth First Search can be used. Their assessment included a speculative examination of typical runtime and a preliminary check of the examination result, for both a Breadth First Search and a Depth First Search utilization. We can also said that Depth First Search is significantly dynamically effective with its memory use, regardless if the Breadth First Search is completed as a duplicated iterative broadening the Depth First Search of memory use can be onpar with Depth First Search and the principle cost is a punishment for the runtime [9].

A. Graph databases

Various databases store data in various manners. Social databases store data in relation as substances and properties, of the elements of tuples and the qualities as segments. The database of the Graph stores the substances as nodes, the edges as connections of the elements of a graph. This informations are associated and creates it simpler for crossing the associations, for instance discovering data, for example, "companion of-a-companion of-a-companion" of the social media network[10]. Apart from this, the more examples of regularly associated information are buy history, where an individual is connected to its buys, or maps with streets and crossing points.

This sort of capacity is especially helpful when displaying information with center around the connections between substances in the structure since they are effectively found and navigated, in distinction to how they are put away in a relational database.

The queries derived from the Cypher can be isolated to a few classes relying upon to the properties, and the different properties are consolidated and termed as the classes. Some separate questions can be created regarding the fixed length and variable length of the properties. The limitations of the length are upper, lower, both also confines the conceivable way lengths. A new property to separate the arrival esteems, for example, paths, nodes or unmistakable nodes.

B. Basic Search Algorithms

The Breadth First Search is a graph search algorithm that begins in a single hub and searches one profundity, by level, at a time. This implies the main level, LEVEL1, is made out of the considerable number of neighbors to the source hub. The following level, LEVELi, are developed of all neighbors to the past level, LEVELi-1, that don't available in other forms of the past levels. When a way to the objective node is discovered, the algorithm closes and there is no experience of a given most extreme profundity is coming to. The given graph is navigated as broad as conceivable sole goes further when all nodes in the depth and flow profundity have been looked.

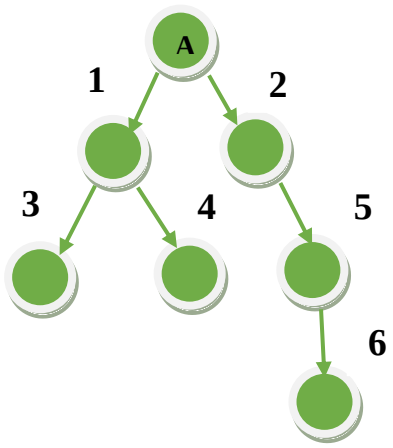


Fig.1 The search pattern for a BFS from the A node.

This example can be found with figure 1, here the inquiry begins with the initial node A and travels the nodes in the indicated request. Execution of the Breadth First Search algorithm is somewhat extraordinary as it enables numerous ways to a similar node, due to the need to discover all ways, yet else it treated as equivalent. If there is any possibilities to traverse same edges then nodes can be rehashed.

Another search algorithm is Depth-First Search which, rather than first investigating all neighbors of the source node A, takes the main defeat and tails it to the neighbor B. It's at that point pursues the principal edge from B and proceeds thusly until a hub with no new edges to investigate is coming to. The algorithm backtracks in this point upto it arrives at a node, and goes down in a similar way with unexplored edges. The following graph originally navigated as profoundly as could reasonably be expected, before withdrawing just it required. The example which is appeared as in Figure 2 is a pursuit from node A. The graph is the ebb and flow search calculation of the Neo4j execution also a few advancements and varieties available.

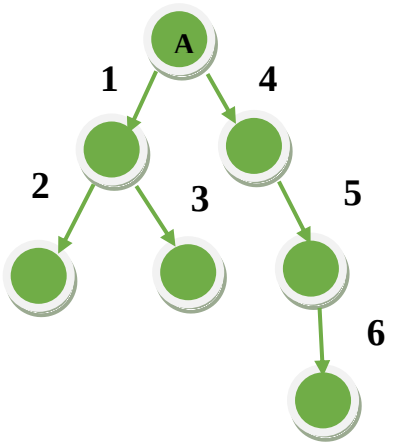


Fig.2 The search method of a Depth First Search from node A.

Figures indicating the quest designs for the distinctive algorithms. Numbers shows the request the algorithm looks through the nodes.

C. Benchmarks

To have the option to make a solid assessment, benchmarks will be utilized. Benchmarks have been utilized since having the option to make a similar quest for both Breadth First Search and Depth First Search without change is important, and having the option to re-try the problem requests with similar outcomes. A broad assortment of the benchmarks can be utilized to cause the organizer to pick the ideal inquiry graph. Genuine datasets are utilized since it is hard to make a created dataset be actually similar to a genuine one. To get a precise estimate since the occasions won't be the very same unfailing, the inquiries will be run on various occasion. Getting the outcomes significantly increasingly exact the inquiries will run with warm-up and the occasions for this won't be remembered for the outcomes. All these are s done with the goal by the machine which doesn't has void stores and work for certifiable use.

A few questions and data set values are available in the LDBC informal social media benchmark[1]. These are the data sets of a dependable and confided in the applications of benchmark of a given graph databases. At the end clients gives situations which it can be perceive; for merchants the agendas are includes the execution of attributes, and also for scientists it will give intriguing situations which will test of the most graph database utilization [12]. The LDBC information generator can create distinctively measured models of a genuine interpersonal, organization utilizing diversing measurement factors, which we utilized measurement factor 1 (SF01) and measurement factor 10 (SF10).

Interpersonal organization is a benchmark that is based on use cases from certifiable use. The utilized questions from this benchmark, are

littler. They were utilized in light of the fact that they are from this present reality and they are moderately straightforward questions on a bigger graph. To begin the work, the ebb and flow Depth-First Search (DFS) execution just as the couple of employments of Breadth-First Search (BFS) in the Neo4j usage were considered. The operands which were further incorporated with Neo4j and an assessment of the operand exhibition attributes were finished. The after effect of the attribute assessments are contrasted with the Depth First Search execution performence on similar inquiries and databases. Dependent on the correlation a hypothetical model was calculated by best performance. The calculated model may further be utilized to the inquiry organizer for picking the pursuit technique.

D. Method

The Breadth First Search operand which uses a non-enhanced Breadth-First Search method from one initial hub in the ordinary case. This pursuit discovers all the ways, of the shortest, from the initial node to other reachable nodes available in the database. Subsequently, the search was sifted with the goal that the ways end in one of the objective nodes. It doesn't discover only the most limited way since that was regarded as the reason for the operand, the method is done additionally as the current Depth First Search operand does on account of the same edge traversal. In general case, on account of the confinement all the ways ought to be discovered, improvements with respect to the most limited way can't be actualized.

The usage section was lead with test-driven improvement for ensuring the code from the earliest starting point and also additionally ensured the modification of the code that was not modified during the actualization of different areas. This method is likewise evidence of our answer filled in the proposed method.

Towards, initialization of usage, we create an insight for the organizer, compelling the utilization of the operand. This was done to ensure the organizer was not modified for different executions and it was simpler to ensure that the operand was actualized accurately. The insight was gotten by the parser and later an alternative was added to have the option to indicate which hub to begin the pursuit from. This was accomplished for testing purposes. Subsequently, all the channeling stages between the operand and the organizer were actualized. At the point when this was done our operand was actualized. At the point when our standard BFS was done, advancements were actualized also. For every single new execution experiments was built to ensure that the usefulness was right.

We directed an assessment at this point when our operand was actualized. The assessment is among the customary non enhanced, the distinctive streamlined Breadth First Search just for the same of ebb and flow search calculation in Neo4j execution of Depth-First Search. It utilized the outcomes from the benchmarks, and oneself developed experiments depending on the graphs. The assessment depended of consequences in runtime and memory use. We built a hypothetical model that figures out which scan strategy ought to be utilized for the hypothetically quickest reaction by this assesment. The arrangement of this hypothetical model can sooner or later executed by Neo4j, ideally, increases the speed of questioning.

E. Implementation

To discover all the ways coordinating an example, the execution is finished by actualizing an algorithm that uses a Breadth First Search. It was actualized as an ordinary Breadth First Search yet without sparing the nodes that had been visited. This is a result for all inquiries, all ways must be returned and subsequently nodes can be visited ordinarily from various nodes. On the off chance that an inquiry indicate that not all ways ought to be restored the outcomes are separated after the hunt is done, since this is the way the previous code and the organizer were planned. Our Breadth First Search operand utilizes a line which holds a hub and the connections are initiates by the source node.

F. Setup for the Experiment

The Breadth-First Search and the previous Depth-First Search on the inquiries of the assessment was finished by utilizing the various adaptation methods. The above methos gives a decent correlation of the speed of the pursuits, which does not utilize the memory. For checking the memory utilization some experiments are build and deliberate the memory use and the runtime. The above experiments are based on the databases of graphs and the runtimes of the benchmarks which were estimated by Neo4j's benchmark apparatuses send to us after calculations. These occasions were mean occasions of the calculatrions and some different measurements that we didn't generally utilize. For the experiments nonetheless, we needed to quantify the occasions and memory utilization with in us.

The runtime was finished by checking the framework time before the run and afterward a short time later and keeping the distinction. For the memory utilization, a technique existed that gave the assigned bytes. We utilized this strategy similarly as the time estimation. The two correlations were weighted together, by physically looking at the outcomes between operands, to show signs of improvement assessment. Since the cost effectiveness can't be dictated, the signs shows the utilizing speed and memory utilization.

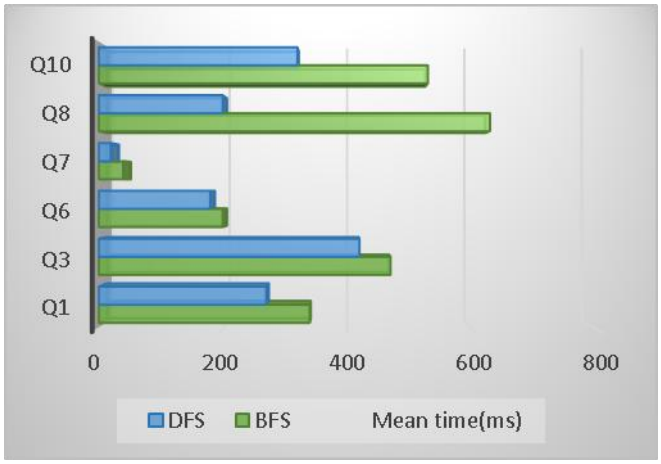
G. LDBC

The two algorithms were ran on two distinct PCs, one is the normal Breadth First Search forms and the another one is Depth First Search, anyway the two projects contained a similar codings. The outcomes for running two normal Breadth First Searcg and the Depth First Search are found in figure 3.a and showed the inquiries of our Breadth First Search arrangement with no improvement was somewhat more slow compared to the Depth First Search algorithm.

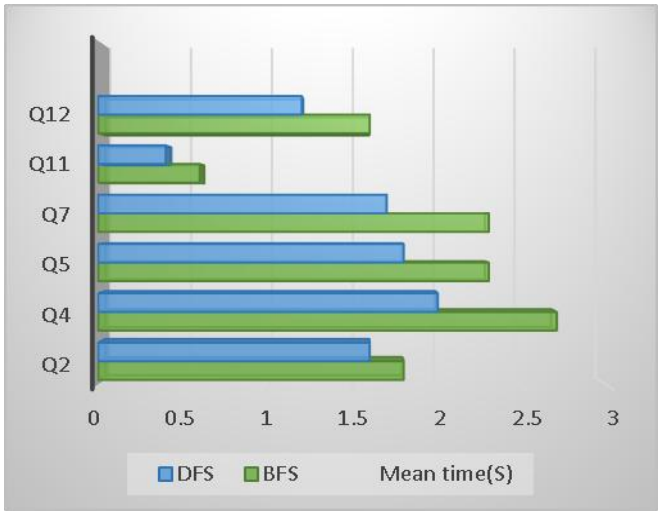
The query 8 and 10 of our Breadth First Search was generously more slow, anyway utilizing the FROM indication and determining which hub to begin looking by the improved runtime. The Question 8, of our Breadth First Search operand was multiple times more slower comparitively to the Depth First Search and when utilizing the FROM insight it was just multiple times more slower than Depth First Search, found as Figure 3.a. Utilizing the

FROM clue was in every case quicker than ordinary Breadth First Search and it was additionally quicker compared to the Depth First Search for Query 5 and 10 and for Short Query 4. From the Figure 3.b, we see the Short Query 7 that utilizing the FROM clue for our Breadth First Search was 23% quicker than our Breadth First Search without the indication and 32% quicker comparatively to the Depth First Search. Depth First Search was quicker compared to our Breadth First Search for the unaltered greatest length of the graphs in the ways.

In our outcomes, a similar example, could be viewed as in the investigation by J. Cobleigh, L. Clarke and L. Osterweil [7], the Depth First Search was regularly quicker than the Breadth First Search for investigation. This could, to a limited extent, be a direct result of the path that the quality of Breadth First Search is most brief way, the primary way it finds will consistently be the shortest. In correlation, the Depth First Search needs checking of all ways between two nodes to check whether they are comparatively less than the effectively discovered most limited path between them.



(a) Long Queries



(b) Short Queries

Fig.3 The local runs of the LDBC benchmark with mean times, in ms and s. In (a) and (b) the results were found from more complex and longer queries.

This implies, for the shortest path issue, the Depth First Search can search the other path of the briefest way and waste runtime. At the point when all ways must be visited this quality is expelled from our BFS, in this way backing it off. This may prompt that the Breadth First Search operand is comparatively slower than the Depth First Search.

Query		BFS	DFS
p = (n0)-[*0..3]-()-[*0..3]-(n1) p = (n0)-[*0..6]-(n1) n0 is "3s" and n1 is "6t" With 10 nodes	Memory (bytes)	3,337,985 3,389,049	3,013,513 3,013,513
	Time (ms)	24 22	14 14
p = (n0)-[*0..3]-()-[*0..3]-(n1) p = (n0)-[*0..6]-(n1) n0 is "24s" and n1 is "74t" With 100 nodes	Memory (bytes)	87,837,001 87,342,433	85,059,577 83,321,713
	Time (ms)	270 159	104 75
p = (n0)-[*0..3]-()-[*0..3]-(n1) p = (n0)-[*0..6]-(n1) n0 is "25s" and n1 is "75t" With 1,000 nodes	Memory (bytes)	656,077,801 656,077,561	651,211,655 651,211,655
	Time (ms)	1,133 513	490 475
p = (:S)-[*0..5]-()-[*0..4]-(:T) p = (:S)-[*0..9]-(:T) S = {5}, T = {1,11} With 16 nodes	Memory (bytes)	195,164,801 196,064,129	191,996,857 192,823,265
	Time (ms)	370 219	157 174
p = (:S)-[*0..5]-()-[*0..5]-(:T) p = (:S)-[*0..10]-(:T) S = {5,15,25}, T = {1,11,21} With 26 nodes	Memory (bytes)	2,794,004,305 2,794,004,305	1,213,247,793 1,212,892,833
	Time (ms)	2,076 2,035	999 1,084
p = (:S)-[*0..7]-()-[*0..7]-(:T) p = (:S)-[*0..14]-(:T) S = {5,21,34,200}, T = {0,13,29,206} With 400 nodes	Memory (bytes)	75,864,000,065 75,864,000,065	75,401,728,633 75,401,149,257
	Time (ms)	120,959 120,759	54,527 54,182

Table 1. Table gives the details of memory usage in bytes and runtime in ms of some nodes. The source and target nodes were specified and every quetion has two MATCH clauses, the top one is for the BFS and the next one is for DFS.

V. CONCLUSION

Let ST is a spanning tree developed by the center point based guess algorithm. ST treated as a Depth First Search tree established by the inside cent, and the tallness of T is never surpass root(G). For clearing T, we apply a searcher on cent. At that point, we recursively clear the branch at cent, with regarding cent as a center point. Consequently, to vanish a part of stature root(G), we have utilized f(root(G)) = f(root(G)/2)+1 searchers. By unraveling repeat, we acquire $s(T) = f(root(G)) = O(\log root(G))$. In addition, $s(T^*) \geq 2$ at whatever point $|VER(G)| > 1$. This pursues that $s(ST)/s(T^*) \leq O(\log root(G))/2 \leq O(\log n)$.

This paper presented an issue that is a blend of spanning tree with node looking, the fundamental variation of graph searching. These issue is called hub looking spanning tree issue. We taken a spanning tree whose search number is at the base on this issue. For selecting a spanning tree ST of G we must check that $s(ST) \leq k$. Furthermore, to adapt to the NP-hardness for the node looking spanning tree issue, we proposed an O(log n)- estimation algorithm. From Neo4j database sets examine process it is demonstrated that Depth First Search is quicker than Breadth First Search. In view of this outcome we can infer that Deapth First Search based node search with spanning tree is superior to the Breadth First Search technique.

REFERENCES

1. O. Erling, A. Averbuch, J. Larriba-Pey, H. Chafi, A. Gubichev, A. Prat, M.D. Pham, and P. Boncz. The LDBC Social Network Benchmark: Interactive Workload. In SIGMOD '15: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, pages 619–630, 2015.
2. J.B. Kruskal, On the shortest spanning subtree of a graph and the traveling salesman problem, Proceedings of the American Mathematical Society 7(1956) 48-50.
3. R.C. Prim, Shortest connection networks and some generalizations, Bell System Technology Journal 36(1957) 1389-1401.
4. M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman & Co., New York, NY, 1990.
5. Nopadon Juneam1, Ondrej Navrátil2, Sheng-Lung Peng. "On the Node Searching Spanning Tree Problem)", Journal of Computers Vol. 29 No. 1, 2018, pp. 160-165.
6. Alexander Olsson, Therese Magnusson." Implementing and Evaluating a Breadth-First Search in Cypher", ISSN 1650-2884 LU-CS-EX 2018-16.
7. J.M. Cobleigh, L.A. Clarke, and L.J. Ostenveil. The Right Algorithm at the Right Time: Comparing Data Flow Analysis Algorithms for Finite State Verification. In Proceedings of the 23rd International Conference on Software Engineering. ICSE 2001,pages37–46,May2001.
8. J.Hipp, U.Güntzer, and G.Nakhaeizadeh. Algorithms for Association Rule Mining – a General Survey and Comparison. SIGKDD Explor. Newsl., 2(1):58–64, June 2000.
9. T. Everitt and M. Hutter. A Topological Approach to Meta-heuristics: Analytical Results on the BFS vs. DFS Algorithm Selection Problem. CoRR, abs/ 1509.02709, 2015.
10. I. Robinson, J. Webber, and E. Eifrem. Graph Databases. O'Reilly Media, Inc., 2013.
11. J.J.Miller. Graph Database Applications and Concepts with Neo4j. In Proceedings of the Southern Association for Information Systems Conference, pages 141–147, Atlanta, GA, USA, March2013.
12. LDBC, The Graph & Benchmark Reference. https://github.com/ldbc/ldbc_snb_docs. Last retrieved 13 March 2018.

AUTHORS PROFILE

Ms. D.JASMINE PRISKILLA pursued Bachelor of Science from University of Madras, in 1998. Master of Science from Bharathidasan University in 2000 and Master of Philosophy in 2002 from Mother Teresa University. She is currently working as Assistant Professor in Department of Computer Science and Computer Applications, Adhiyaman Arts and Science College for Women, Uthangarai, Tamilnadu, India. Her main research interest is in the area of Algorithms, Data Structures and Programming Languages. She has 19 years of teaching experience and 14 years of research experience.

Dr. K. ARULANANDAM is currently working as Assistant Professor & Head in the Department of Computer Science and Applications, Government Thirumagal Mills College, Gudiyattam. He has published several papers in reputed National and International journals. He has 17 years experience in teaching and research experience.