Journal of Advanced Computing Research Vol. 1, Issue 1 (2016) 31-34

Recognizing Unordered Depth-First Search Trees of an Undirected Graph in Parallel

Sharifah Nursyuhada binti Syed Zulkafli [#1], Mohamed Faidz Mohamed Said [#2]

[#] *Universiti Teknologi MARA*
70300 Seremban, Negeri Sembilan, MALAYSIA

[1] syhdzkfli@gmail.com

[2] faidzms@ieee.org

***Abstract*—Depth First Search (DFS) is surely understood by the imperative procedure to planning successive calculations at charts. DFS systems can be parallelized as the one that can be expected, a great deal of successive diagram calculations should also be possible. In this study, assumption of P as an undirected diagram and W as a traversing tree of P. An effective parallel calculation is introduced for figuring out if W is an unordered profundity first pursuit branch of W. The proposed calculation keeps running in O (n/q + log n) period utilizing processors on the EREW PRAM, where n is the quantity of edges contained in G. It is cost-ideal and accomplishes direct speed.**

***Keywords*: DFS branch, spanning trees, parallel algorithms, PRAM**

I. INTRODUCTION

Depth First Search is well known as DFS, is surely understood by the imperative procedure to planning successive calculations at charts [13]. DFS systems can be parallelized as the one that can be expected, a great deal of successive diagram calculations should be possible also. Lamentably, paper [10] demonstrated that it appears to be difficult to check effectively in parallel whether a given request of vertices is equivalent to the meeting arrangement acquired by playing out a requested DFS at a chart, to infer that requested DFS is intrinsically consecutive. Accordingly, numerous specialists swung their attention to other related themes. At the point when the requested confinement is expelled, the positive feedback can be known. Paper [5] introduced the O (log n) period parallel unordered DFS calculation on planar undirected diagrams. Paper [1] introduced the randomized NC calculation for performing unordered DFSs on general coordinated diagrams. Inverse to the development of a DFS branch, paper [11] determined the issue by figuring out if a given spreading over branch of a coordinated diagram is an unordered DFS branch of the chart. It also demonstrated the difficult that can be understood in *(log^2 m)* period [11].

The issue for inspect in order to give traversing branch of an undirected outline is an unordered DFS branch of the graph is portrayed [6]. This demonstrates undirected chart containing vertices *v* and *d(≥m-1)* edges, its unordered DFS branch can be perceived in *U(M/P + log m)* period utilizing *p* on the *EREW PRAM.*

Issue that confirming in order to give spanning tree fulfills some particular properties is of hypothetical interest. Consequently, the issue of perceiving different spreading over branch had been broadly examined in written works. For instance, other than DFS branch, paper [8] concentrated on the issue of perceiving expansiveness first pursuit trees, papers [14] and [2] examined the issue of perceiving least traversing branch, and considered the issue of perceiving most brief way branch [9]. An efficient algorithm for perceiving DFS branch has a few applications [7], [11]. For instance, in paper [11], that was specified that an effective calculation for perceiving DFS branch can be utilized as a subroutine for a calculation that develops a DFS branch by progressively producing hopefuls until a legitimate one is acquired. Two illustrations were given by the researcher [7]. Consider an undirected chart G in which no two edges has the same weight. Other than being of hypothetical intrigue, the acknowledgment issue of DFS branch is additionally of down to earth significance. In this present reality, a calculation situation is not generally dependable. Consequently, that is important to confirm the yields of a DFS branch development calculation or to make sure the correctness of a DFS branch inputted into a method.

Let the *Q = (R,S)* is an undirected diagram made out of |S| = *k vertices* and |*E*| = *d(≥m-1)* edges. An unordered depth-first search branch of J is an established spreading over branch of J yield by playing out the accompanying nondeterministic DFS calculation.

Finding 1 (Unordered depth-first search)

In: An undirected connected graph F.
Out: An unordered DFS branch Z of F.

A. Select a vertex rv as the starting point.
B. Call DFS(rv).

Step for DFS *(rv)*

I. Point the rv as the visited.
II. Every vertex *p* is together with rv
III. If and only the mark(rv,p) was not visited then it will call as an edge of *Z* and call DFS(p)

Firstly, step A is chosen and it is dealt of the base yield DFS tree. As known that Steps A and II are nondeterministic, it might be more unordered DFS branch is shown. The *perceive an unordered* DFS branch is to figure out if a traversing branch is a conceivable yield of the under unordered profundity first hunt calculation, choose a meeting request. Truth be told, if Z is known not an unordered DFS branch of diagram G, the meeting request that will infer by playing out a preorder traversal on *Q* utilizing the calculation [3].

Steps A and II are nondeterminism made the affirmation issue perplexed. Two phases are accepted deterministic, illustration that a specific crest c will relegated as an establishment of *Z*, and for each vertex *r* we cross the abutting vertices of *v* taking after the solicitation of the suggested proximity once-over of it. By then, to figure out if *Z* is a DFS branch or not can be basically done in straight period using the standard significance first chase computation. Make sure that if the two phases are deterministic, the got DFS branch is asked. In case solitary step II is nondeterministic and a specific vertex r is allotted as the establishment of Q, check adequately by using the famous characteristic of DFS branch: Q is a DFS branch if and just if Q do not have the cross edge [11].

For the circumstances that have both Steps A and II are nondeterministic, an immediate period progressive figuring. Korach and Ostfeld [7] are the individuals that presented seeing unordered DFS trees. An acknowledgment issue of the instance of coordinated charts is the characterized comparably. An acknowledgment issue based on coordinated diagrams is difficult than on undirected charts, in light of the fact that an undirected chart can be effectively changed over into a guided one by supplanting each undirected edge (rv,p) with two coordinated edges (rv,p) and also (rv,p) and after that can be unraveled by utilizing the calculations for coordinated charts. Schevon and Vitter demonstrated that the acknowledgment of unordered DFS branch for coordinated charts should possible in *(log^2 m)* period using *U(n^2376)* processors on a CREW PRAM.

In the coordinated case, there are only peaks of the in degree 0 in the coordinated crossing branch *Z*, and therefore the root is constantly assigned. In this study, it also demonstrates that an acknowledgment of undirected charts without an assigned root should be possible in *U (n/q + log m)* parallel period utilizing p processors on the *EREW PRAM*. The real strategy used in the calculations is the Euler tour method [4], [12], [15], that is understood becoming a decent worldview of outlining productive parallel calculations.

II. A NECESSARY AND SUFFICIENT CONDITION FOR RECOGNIZING DFS TREES

Let $Q = (R,S)$ be an undirected diagram made out of, $|S| = k$ vertices and $|E| = d(\geq m-1)$ edges and Q be a spreading over tree of Z.

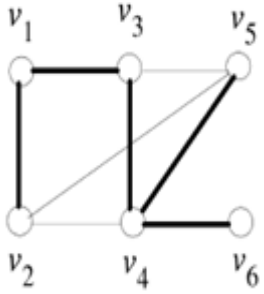

Figure 1. A spanning tree Z of a graph F

The branch edges are just a simple path that comes from a branch path. Knowing that branch way interfacing any two vertices v and c is one of a kind, it can be known meant as branch path (i,c). The crossing tree *Q* is a free branch, that is, there will be no root assigned. At the point when a root r is relegated for *Q*, the spreading over branch will be signified as *Q*®.

Overall the case, all non-tree edges will be ordered into two types that are cross edges and back edges. A non-tree edge (i,c) is known as a cross edge if *i* and *c* are not progenitors to each other; else, it can be known as a back edge. Surely it can be understood that Q® is a DFS branch of *G* if and just if there do not have cross edge, or proportionally, all the non-tree edges are back edges [11]. For instance, reflect the branch *Z* and the chart *G* portrayed in Figure 1 In the event that chooses v1 as the root, it is not DFS branch because it has two types of edges that is (v2,v4) and (v2,v5). Then again, in the event that choose v2 as the root, *Q*(v2) is a DFS branch because no cross edge was found in the branch. In this study issue, the base is found spreading over branch Q that not assigned. So that vertex r*v* of *Q* can be the root that is Q***(v)*** is a DFS branch, it is important to check for each vertex *v* whether all non-tree edges are back edges regarding ***Q(v)***. Such a vertex *v* is known as a competitor base of Q. We ought to know about that the expressions "cross edge" and "back edge" are important just for an established tree. The back edge can be said as other root, meanwhile a non-branch edge can be a cross edge.

Theorem A

Let $e = (v, w)$ be a cross edge with respect to $T_{(r)}$. Then, $CROSS(e) = V - (X_v \cup X_w)$ and

$$BACK(e) = X_v \cup X_w,$$

where X_v and X_w denote the two subtrees in $T_{(r)}$ rooted at v and w, respectively.

Theorem B

Let $e = (v, w)$ be a back edge with respect to $T_{(r)}$ and v be an ancestor of w. Then, $CROSS(e) = Xchild(v, w) - X_w$ and $BACK(e) = V - (Xchild(v, w) - X_w)$, where $Xchild(v, w)$ and X_w denote the two subtrees in $T_{(r)}$ rooted at $child(v, w)$ and w, respectively.

III. A SEQUENTIAL RECOGNITION ALGORITHM

In this segment, the Euler-visit was applied in the issue and consecutive calculation was calculated to determine the accuracy of the methodology. Successive calculation will be effectively parallelized the most.

Let *Q = (R, S)* be an undirected graph and *Q* be and spanning branch of *Z*. First, determine an arbitrary vertex *b* in *S* and orient *Q* into a rooted branch *Q*® Then, input the set *H = S-∪(e ∈E -Q) CROSS(e*) by initially setting *H*. The *S* and then pruning away from *U* the vertices in *CROSS*. *e* for every non-branch edge *e ∈E – Q*.

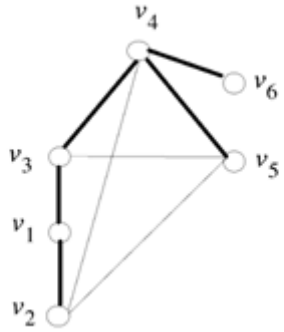

Figure 2. A spanning tree Z of a graph Q

In this study, it determines the branch *Z* and the graph Q is depicted in Figure 2. The *v4* was assumed as the root of *Z*. Firstly, set *H = (v1, v2 . . . v6).* There are three non-branch edges, i.e. *(v2, v4, v2, v5, and v3, v5).* According to Theorem B, vertices *v1* and *v3* are pruned away from *H* for the non-tree edge *(v2, v4)* since they are in the sub tree rooted at *v3* child (*v4, v2*) but not in the sub branch rooted at *v2*. According to Theorem A, vertices *v1, v3, v4*, and *v6* are pruned away from *H* for the non-branch edge *(v2, v5*), therefore *v2* and *v5* are not the two sub branch respectively. Similarly, vertices *v4* and *v6* are pruned away from *H* for the non-tree edge (*v3, v5*), therefore *v3* and *v5* are not the two sub branch respectively. That only vertices remain in the branch after the pruning is *v2* and *v5*.

Clearly to perform Theorem A and B effectively, ought to have the capacity to determine if a peak is inside the sub branch and set up at another peak. By prudence of this, Euler visits can be exceptionally useful. For an established branch *Q*®, an Euler visit can coordinated way beginning and closure at *k* and navigating every branch edge forward and in reverse precisely once. An Euler visit for the established branch Z (v4) portrayed in Figure 2 is

v4 →v3→v1→v2→v3→v4→v5→v4→v6→v4

An Euler circuit can be built, basically put on a profundity first inquiry on the given branch Q®, and record the succession of went to edges. An Euler visit is typically spoken to by an arrangement of vertices inter weaved with bolt signs. As a result of the meaning of an Euler visit, the data about all sub branch of Q® is put away in its Euler visit. All in all, the sub branch established at a peak in Q® compares to the subsequence encased by the first and the last events of the peak in the Euler visit. On the off chance that a vertex is a leaf in Q®, then it happens in the Euler visit precisely once and its sub branch is this vertex itself. In view of the above property,

Theorem A and B can be effortlessly executed by utilizing Euler visits. From the above talk, whether *Q* is a DFS branch of *G* can be perceived utilizing the accompanying strides. To determine *Q* is as DFS branch of *G* can be shown by using some tips.

1. The arbitrary peak *k* is selected and situates *Q* to the established branch Q®.
2. An Euler visit *U* of Q® is developed. Note that the arrangement of vertices in *U* is equivalent to *V*
3. An each cross edge *s = (i,c)* as for Q®, prune far from U the vertices outside the two subsequences relating to the two sub branch established at v and c, separately.
4. For each back edge *s = (i,c)* as for Q®, expecting v is a predecessor of w, first find child (i,c), and after that prune far from U the vertices that compare to the sub branch established at child (i,c), aside from the vertices that relate to the sub branch established at *w*.

It might be P(a) can be pruned by the vertices of each non-branch edge and there are P(b) non-branch edges, a direct execution will take P(ab) time. To decrease the time many-sided quality, the vertex-pruning methodology will be actualized by a prefix entirety calculation along the Euler visit. It will allocate +1 to the coordinated edge before *vi*, -1 to the coordinated edge after *vj*, and 0 to all other coordinated edges and vertices. After a prefix aggregate operation is played out, the prefix aggregates of the vertices in this subsequence are 1, and the prefix aggregates of different vertices are 0, which is shown as takes after. For a case in point, refer to the accompanying chart. On the off chance that the subsequence of vertices from vi to *vk* ought to be pruned, we allocate +1 to the coordinated edge before vi and -1 to the one after *vk*. On the off chance that the subsequence of vertices is another from vj to versus to be prune, +1 and also -1 will be allocated to the coordinated edges before vj and after versus, separately. For the various coordinated edges and vertices, allocate 0. The prefix aggregate will be performed and the vertices with prefix that more than 0 are prune away.

IV. CONCLUSION

Depth First Search systems can be parallelized as the one that can be expected, a great deal of successive diagram calculations should also be possible. In this study, an undirected diagram and a traversing tree has been shown. An effective parallel calculation is introduced, and the proposed calculation keeps running in O (n/q + log n) period utilizing processors, where n is the quantity of edges and it is cost-ideal and accomplishes direct speed. In the sequential recognition algorithm segment, the Euler-visit was applied, and consecutive calculation was made to determine the accuracy of the methodology. Successive calculation will be most effectively parallelized.

REFERENCES

[1] A. Aggarwal, R.J. Anerson, and M.-Y. Kao, "Parallel Depth-FirstSearch in General Directed Graphs," SIAM J. Computing, vol. 19, no. 2, pp. 397-409, 1990.

[2] B. Chazelle, "Computing on a Free Tree Via Complexity-Preserving Mapping," Algorithmica, vol. 3, pp. 337-361, 1987.

[3] C.C.-Y. Chen, S.K. Das, and S.G. Akl, "A Unified Approach to Parallel Depth-First Traversals of General Trees," Information Processing Letters, vol. 38, pp. 49-55, 1991.

[4] U. Vishkin, "On Efficient Parallel Strong Orientation," Information.

[5] T. Hagerup, "Planar Depth-First Search in O.log n. Parallel Time,"SIAM J. Computing, vol 19, no. 4, pp. 678-704, 1990.

[6] J. JaJa, An Introduction to Parallel Algorithms. Addison Wesley, 1992.

[7] E. Korach and Z. Ostfeld, "DFS Tree Construction: Algorithms and Characterizations," Proc. 14th Int'l Workshop Graph-Theoretic Concepts in Computer Science (WG-88), 1988.

[8] U. Manber, "Recognizing Breadth-First Search Trees in Linear Time," Information Processing Letters, vol. 34, pp. 167-171, 1990.

[9] C.-H. Peng, J.-S. Wang, and R.C.-T. Lee, "Recognizing Shortest- Path Trees in Linear Time," Information Processing Letters, vol. 57, pp. 78-85, 1994.

[10] J.H. Reif, "Depth-First Search Is Inherently Sequential," Information Processing Letters, vol. 20, pp. 229-234, 1985.

[11] C.A. Schevon and J.S. Vitter, "A Parallel Algorithm for Recognizing Unordered Depth-First Search," Information Processing Letters, vol. 28, pp. 105-110, 1988.

[12] B. Schieber and U. Vishkin, "On Finding Lowest Common Ancestors: Simplification and Parallelization," SIAM J. Computing, vol. 17, pp. 1,253-1,262, 1988.

[13] R.E. Tarjan, "Depth-First Search and Linear Graph Algorithms," SIAM J. Computing, vol. 1, pp. 814-830, 1972.

[14] R.E. Tarjan, "Applications of Path Compression on Balanced Trees," J. ACM, vol. 26, pp. 690-715, 1979.

[15] R.E. Tarjan and U. Vishkin, "An Efficient Parallel Biconnectivity Algorithm," SIAM J. Computing, vol. 14, pp. 862-874, 1985.