International Journal of Research in Advent Technology, Vol.7, No.4, April 2019
E-ISSN: 2321-9637
Available online at www.ijrat.org

An Optimal Algorithm to Find Next-to-Shortest Path between Two Vertices of Cactus Graphs

Sambhu Charan Barman[1], Madhumangal Pal[2] and Sukumar Mondal[3]
[1]*Department of Mathematics, Shahid Matangini Hazra Govt. College for Women, Nimtouri, Tamluk, Purba Medinipur, India.*
Email: barman.sambhu@gmail.com
[2]*Department of Applied Mathematics with Oceanology and Computer Programming, Vidyasagar University, Midnapore - 721 102, India.*
Email: mmpalvu@gmail.com
[3]*Department of Mathematics, Raja N. L. Khan Women's College, Gope Palace, Midnapore - 721 102, India.*
Email: sm5971@rediffmail.com

Abstract: A cactus graph is a connected graph in which every block is either an edge or a cycle. The next-to-shortest path between two vertices u and v is a path whose length is minimum among all paths between u and v with the shortest ones excluded. In this paper, we present an optimal algorithm to find a next-to-shortest path between any two vertices of a cactus graph with n vertices which runs in $O(n)$ time.

Keywords: Design of algorithms, analysis of algorithms, shortest path, next-to-shortest path and cactus graphs.
AMS Mathematics Subject Classification (2010): 05C30, 68R10, 68Q25

1. INTRODUCTION

1.1. Cactus graphs

Let $G = (V, E)$ be a finite, simple, connected and undirected graph. A vertex u is called a *cut-vertex* if the removal of u and all edges incident to u disconnects the graph. A connected graph without a cut-vertex is called a *non-separable graph*. A *block* of a graph is a maximal non-separable sub graph. In graph theory, a *cactus graph* is a connected graph in which every block is either an edge or a cycle, in other words, every edge in cactus graph belongs to at most one simple cycle. Cactus graph were first studied under the name of the Husimi trees, bestowed on them by Frank Harry and George Eugene Unlenbeck in honour of previous work of these graphs by Kodi Husimi. Cactus graphs have been extensively studied and used as models for many real-world problems. This graph is one of the most useful discrete mathematical structures for modelling problems arising in the real-world. An early application of cactus graphs was associated with the representation of op-amps [25, 26, 27]. Also, the applications of cactus graphs can be found in [15, 19]. Recently cactus graphs have been used in comparative genomics as a way of representing the relationship between different genomes or parts of genomes [29]. This graph is a subclass of planner graph and superclass of tree

.

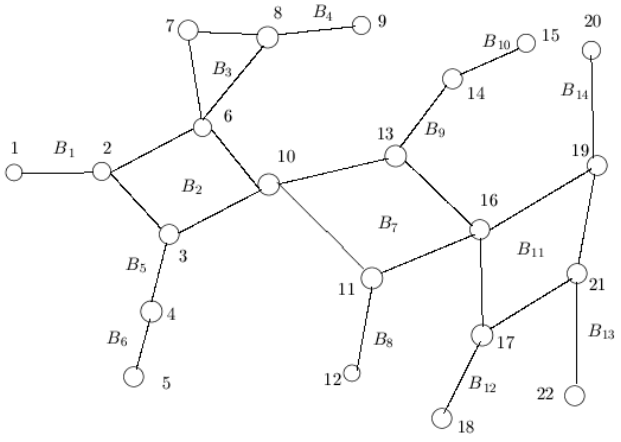

Figure 1. A cactus graph G

1.2. Definitions

Let $G = (V, E)$ be a graph with vertex set V and edge set E, where n is the number of vertices and m is the number of edges. The *distance* between two vertices u and v in G is denoted by $d(u, v)$ and it is the minimum number of edges required to traverse from u to v or v to u. *Next-to-shortest path* between two vertices u and v is a path whose length is minimum among all paths between u and v with the shortest ones excluded. *Next-to-shortest distance* between two vertices u and v is the length of the next-to-shortest path between u and v and it is denoted by $d_n(u, v)$.

1.3. Applications

The next-to-shortest path problem is an important problem in graph theory and it has many applications in real life problems. This problem is a variation of K-shortest paths problem that finds applications in operations research, telecommunications, VLSI design, optimizing compilers for embedded systems [17], etc.

1.4. Survey of the related works

Problem of finding next-to-shortest path between two vertices has received much less attention due to the fact that in directed graphs, when we allow edges of length zero, the problem has been shown to be NP-hard [18]. In [32], Seidel has given an $O(M(n)logn)$ time sequential algorithm for all pair shortest path (APSP) problem for an undirected un-weighted arbitrary graphs with n vertices, $M(n)$ being the time necessary to multiply two $n \times n$ matrices of small integers, the best known time for $M(n)$ is of $O(n^{2.376})$. Mondal et al. [22, 23] have solved APSP problem on permutation graphs and trapezoid graphs in $O(n^2)$ time. Maity and Pal [20] have designed an optimal algorithm to solve APSP problem on cactus graphs. In [31], Saha et al. have designed an optimal algorithm to find APSP on circular-arc graphs. In [8], Eppstein has designed an algorithm for finding the k-shortest paths in a digraph with n vertices and m edges in $O(m + nlogn + k)$ time. In this paper, we focus on cactus graphs. Das et al. have solved many problems on cactus graphs, like maximum weight 2-colour set in [4], minimum dominating set in [5], minimum 2-neighbourhood covering set in [6], etc. Also, Khan et al. [10, 11, 12, 13, 14] have solved edge colouring problem, (2, 1)-total labelling problem, $L(0,1)$-labelling problem, cordial labelling problem and adjacent vertex distinguishing colouring problem on cactus graphs. Finding next-to-shortest paths in undirected graphs with strictly positive edge length in time $O(n^3m)$ is studied [16]. Also Lalgudi et al. [18] have designed an algorithm for computing strictly second shortest paths in directed graphs in $O(n^2)$ time. In [33], Shisheng et al. have designed an improved algorithm for finding next-to-shortest paths in a weighted undirected graph. Mandal et al. [21] have designed an optimal algorithm to solve next-to-shortest path problem on circular-arc graphs, in $O(n)$ time. Recently Barman et al. [1, 2] have designed $O(n^2)$ time algorithm to solve next-to-shortest path problem between two vertices on permutation graphs and trapezoid graphs.

1.5. Main result

In this paper, we present an optimal algorithm to find next-to-shortest path between any two vertices of a cactus graph with n vertices which runs in $O(n)$ time.

1.6. Organization of the paper

In the next section, we discuss about the path(s) between two vertices in a block of cactus graphs. Construction of BFS tree T from cactus graph, main path and some notations related to our problem are presented in Section 3. In Section 4, we prove some important results related to our problem on cactus graphs. The algorithm is presented to find next-to-shortest path between two vertices of cactus graphs in Section 5. The time complexity is also calculated in this section. In Section 6, we present conclusion. Acknowledgement is presented in last Section, i.e. Section 7.

In the next Section, i.e., in section 2, we discuss about the paths between two vertices in a block of cactus graphs.

2. PATH(S) BETWEEN TWO VERTICES IN A BLOCK OF CACTUS GRAPHS

We know that a block of a graph is a maximal non-separable sub graph. Also, a path between two vertices of a simple graph is a finite (for finite graph) or infinite (for infinite graph) sequence of edges which connect a sequence of vertices which are all distinct from one another. Again from the definition of cactus graph, each block of a cactus graph is either an edge or a cycle. For examples, block B_1 is an edge and block B_2 is a cycle of length four in Figure 1. Now we describe the path(s) between two vertices u and v in a block through the following cases.

Case 1: // When block is an edge. //

Let a block contains only two vertices u and v, i.e., the block is an edge. So there is only one path between u and v, which is shortest path. Also, there is no next-to-shortest path between u and v.

Case 2: // When block is a cycle. //

Let $u_1 \mapsto u_2 \mapsto \cdots \mapsto u_n \mapsto u_1$ be a cycle containing n vertices. Also, let two specified vertices be u_1 and u_l. Obviously, there are only two paths between u_1 and u_l, one is $u_1 \mapsto u_2 \mapsto \cdots \mapsto u_{l-1} \mapsto u_l$ of length l and other is $u_1 \mapsto u_n \mapsto u_{n-1} \mapsto \cdots \mapsto u_{l+1} \mapsto u_l$ of length $(n - l)$. Now, there are three subcases may arise, which are discussed below.

Subcase 1: // When $n = l$. //

In this subcase, there are two shortest paths between u_1 and u_l. Also, there is no next-to-shortest path between u_1 and u_l.

Subcase 2: // When $l < n - l$. //
In this subcase, the path of length l is the shortest paths between u_1 and u_l and the remaining path of length $(n - l)$ is the next-to-shortest path between u_1 and u_l.

Subcase 3: // When $l > n - l$. //
Here, the path of length $n - l$ is the shortest paths between u_1 and u_l and the remaining path of length l is the next-to-shortest path between u_1 and u_l.

From the above discussion, we can draw the following conclusions.

- *There is neither alternative shortest path nor next-to-shortest path between two vertices in a block which is an edge in cactus graphs*.

- *In any cycle, there are only two paths between any two vertices, one is shortest path and other is alternative shortest path or next-to-shortest path*.

In the next section, we discuss about the construction of a BFS tree, denoted by T from the cactus graph G.

3. CONSTRUCTION OF BFS TREE *T*

It is well known that the BFS is an important graph traversal technique. Also it constructs a BFS tree. In BFS, started with vertex v as root, we first visit all the neighbours of the root v. Then for each of those neighbours of v, we visit their unvisited neighbour nodes. We keep going until all vertices reachable from the root v have been visited.

To construct a BFS tree from the cactus graph we use a method in which blocks and cut-vertices of the cactus graph $G = (V, E)$ are determined using DFS technique [35, 34] and thereafter form an intersection graph [28] $G' = (V', E')$ where V' is the set of blocks, say $V' = \{B_1, B_2, \dots, B_N\}$ and E' is the set of edges such that $E' = \{(B_i, B_j) : i \neq j \text{ and } B_i, B_j \text{ are adjacent blocks}, i, j = 1, 2, \dots, N\}$. Then we construct a BFS tree, denoted by T with root as a block containing any one among the specified vertices u and v.

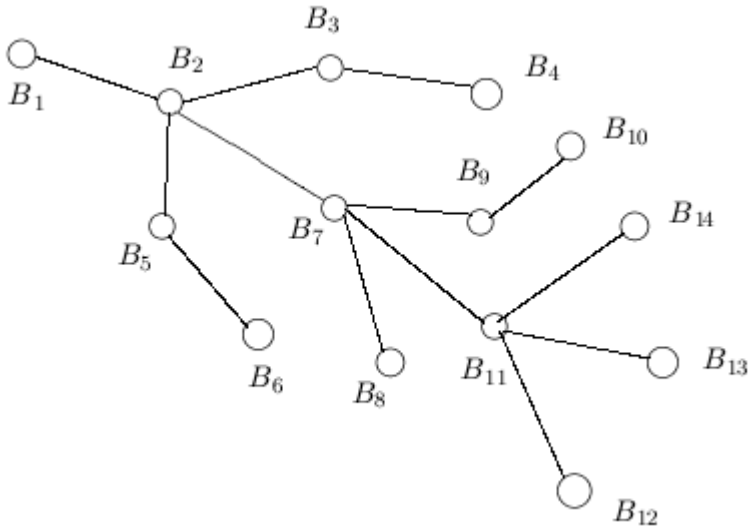

Figure 2. An intersection graph G' of the cactus graph G of Figure 1.

Thus the tree $T = (V', E'')$ where say $V' = \{B_1, B_2, \dots, B_N\}$ and E'' is a subset of E' is computed. For convenience, we refer the vertices of T as nodes. We note that each node of this BFS tree is a block of the graph G. The parent of the node B_i in the tree T will be denoted by $parent(B_i)$. The BFS tree can be constructed on general graphs in $O(n + m)$ time, where n and m represent respectively the number of vertices and number of edges [34]. Also we know that $m = O(n)$ for cactus graphs. So the BFS tree T can be constructed from the cactus graphs in $O(n)$ time.

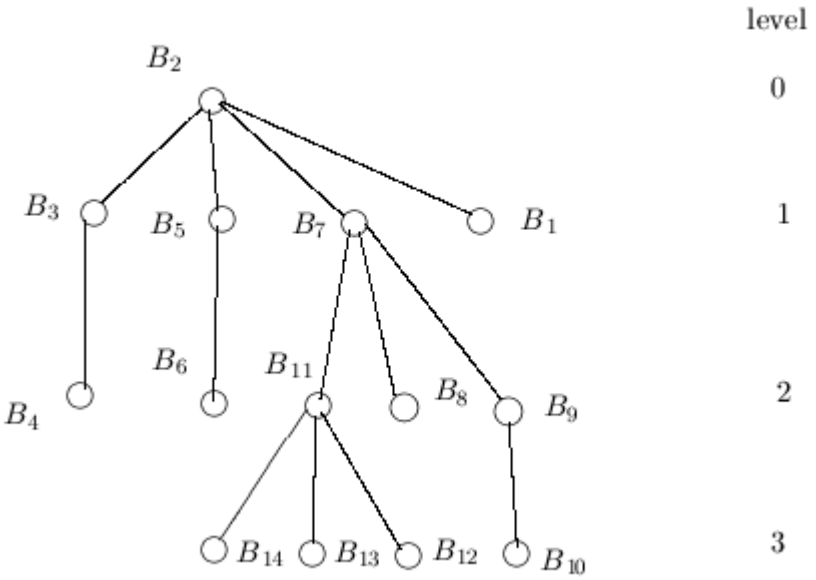

Figure 3. A BFS tree T with root as a block B_2 of graph G' of Figure 2.

The BFS tree T with root as a block B_2 (which contains the specified vertex $u = 2$) of the graph G' of Figure 2 is shown in Figure 3.

3.1. Computation of the main path on BFS tree T

Let the blocks B_i and B_j contain two specified vertices u and v respectively, where $i \neq j$. Now, if we construct a BFS tree T with root as any block among B_i and B_j, then the shortest path between B_i and B_j on T is called the main path. For example, if we construct a BFS tree T with root as B_i then the main path will be $B_j \mapsto parent(B_j) \mapsto parent(parent(B_j)) \mapsto \cdots \mapsto B_i$. For convenience, we denote the node of the main path at level k on tree T is M_k, k = 0,1, 2, …., l where $d(B_i, B_j) = l$ on T. So, the main path is $M_l \mapsto M_{l-1} \mapsto M_{l-2} \mapsto \cdots \mapsto M_0$.

3.2. Some notations

Here we present some notations which are needed to design the algorithm.

Symbol	Description
l_i	l_i is the number of vertices of the block M_i, i = 0,1, 2, …., l.
M_i	M_i is the node of the main path on tree T at level $i, i = 0, 1, 2, \ldots, l$.
e_i	the cut vertex between blocks M_{i-1} and M_i.
$d(u, v)$	shortest distance between the vertices u and v.
$d_n(u, v)$	next-to-shortest distance between the vertices u and v.
M	very large positive integer.
$e_{i-1} \hookrightarrow e_i$	the shortest path between $e_{i-1}$and e_i.
r_i	$r_i = d(e_i, e_{i+1})$.

Before going to our proposed algorithm, we prove the following important results relating to the next-to-shortest paths on cactus graphs.

4. SOME RESULTS RELATED TO THE NEXT-TO-SHORTEST PATHS ON CACTUS GRAPHS

In this section we present some important results related to the next-to-shortest paths on cactus graphs as follows.

Lemma 1*: If u and v are two vertices of a cycle C of length l then $d(u,v) < \left\lfloor \frac{l}{2} \right\rfloor$.*

Proof. To prove this lemma, let $a_1 \mapsto a_2 \mapsto \cdots \mapsto a_l \mapsto a_1$ be an undirected cycle C of length l.

Now, for $i \neq j$ and $i, j \in \{1, 2, \ldots, l\}$,

$$d(a_i, a_j) = \begin{cases} \lfloor l/2 \rfloor, & l \text{ is an odd integer} \\ (l/2), & \text{otherwise} \end{cases}$$

So, $d(u,v) < \left\lfloor \frac{l}{2} \right\rfloor$, for all $u, v \in C$ and $| C | = l$.

Lemma 2*: If C be a cycle of length l and $u, v (\neq u) \in C$ and if $d_n(u,v)$ exists, then* $d_n(u,v) - d(u,v) = l - 2d(u,v)$, *i.e.,* $1 \leq d_n(u,v) - d(u,v) \leq l - 2$.

Proof. Let C be a cycle of length l and $u, v \in C$, where $u \neq v$ and $d_n(u,v)$ exists.

So, $d_n(u,v) + d(u,v) = l$,
or, $d_n(u,v) = l - d(u,v)$,
or, $d_n(u,v) - d(u,v) = l - 2d(u,v)$,

or, $d_n(u,v) - d(u,v) \leq l - 2$ as $d(u,v) \geq 1$.

Again, if $d_n(u,v)$ and $d(u,v)$ both exist then $d_n(u,v)$ and $d(u,v)$ are both integers.

So, $1 \leq d_n(u,v) - d(u,v)$.
Hence, $d_n(u,v) - d(u,v) = l - 2d(u,v)$,
i.e., $1 \leq d_n(u,v) - d(u,v) \leq l - 2$.

5. THE ALGORITHM

To find next-to-shortest path between two specified vertices u, v of a connected cactus graph G, we first construct a BFS tree T from G. After then we mark the main path between block which contains u and the block which contains v and then find the shortest path between u and v. After then we find next-to-shortest path between two specified vertices u and v. If u and v belong to same block of cactus graph then shortest path and next-to-shortest path between u and v are discussed in section

Now we present the main algorithm to find next-to-shortest path between two vertices u and v which are belong to different blocks of a connected cactus graph G is presented following.

Algorithm CNSP

Input: A connected cactus graph $G = (V, E)$.

Output: Next-to-shortest path between u and v.

Initially let the blocks B_i and B_j contain the vertices u and v respectively $i \neq j$.

Step 1. Construct a BFS tree T with root as B_i.

Step 2. Find the main path $M_l \mapsto M_{l-1} \mapsto M_{l-2} \mapsto \cdots \mapsto M_0$ between blocks B_i and B_j, where $(B_i, B_j) = l$ and $u = e_0, v = e_{l+1}$.

Step 3. Find the shortest path between e_{i-1} and e_i from M_{i-1}, for $i = 1, 2, \ldots, (l+1)$.

Step 4. Set $d(e_i, e_{i+1}) = r_i$, for $i = 0, 1, 2, \ldots, l$.

Step 5. Compute $d(u,v) = r_0 + r_1 + \cdots + r_l$ and the shortest path between two vertices u and v is $e_0 \hookrightarrow e_1 \hookrightarrow e_2 \hookrightarrow \cdots \hookrightarrow e_l \hookrightarrow e_{l+1}$.

Step 6. Set $i = 0$.

Step 7. If $l_i = 2$ or $\frac{l_i}{2} = r_i$ then set $p_i = M$ and $i = i + 1$, else set $p_i = l_i - 2r_i$ and $i = i + 1$.

(by **Lemma 2**)

Step 8. If $i = l + 1$ then Stop,

else go to Step 7.

Step 9. Compute $\min\{p_i, i = 0, 1, 2, \ldots, l\} = p_j$ (say).

Step 10. If $p_j = M$ then there is no next-to-shortest path between u and v,

else $d_n(u,v) = d(u,v) + p_j$ and the next-to-shortest path between u

and v is obtained by replacing $e_j \hookrightarrow e_{j+1}$ by the next-to-shortest path between e_j and e_{j+1} from $u \hookrightarrow v$.

End CNSP

Using **Algorithm CNSP** we get the next-to-shortest path between the vertices 2 and 18 is $2 \mapsto 3 \mapsto 10 \mapsto 11 \mapsto 16 \mapsto 19 \mapsto 21 \mapsto 17 \mapsto 18$, whereas $2 \mapsto 3 \mapsto 10 \mapsto 11 \mapsto 16 \mapsto 17 \mapsto 18$ is the shortest path between 2 and 18 for the cactus graph of Figure 1.

5.1. Illustrative Example

To illustrate the execution process of Algorithm CNSP, we consider the vertices $u = 2$ and v = 18 of the cactus graph of Figure 1 as the specified vertices, i.e., we want to find next-to-shortest path between the vertices 2 and 18. Obviously, $2\epsilon\ (B_1 \cap B_2)$ and $18\ \epsilon B_{12}$. We consider 2 as a member of the block B_2. In Step 1, we construct a BFS tree T with root as the block B_2 (see Figure 3). After then, in Step 2, we compute the main path which is $M_3(=B_{12}) \mapsto M_2(=B_{11}) \mapsto M_1(=B_7) \mapsto M_0(=B_2)$, where $d(B_2, B_{12}) = l = 3$. Also, we take $u = e_0$ and $v = e_{l+1} = e_4$. Also, from Figure 1, $e_1 = 10$, $e_2 = 16$ and $e_3 = 17$ are the cut-vertices between the blocks M_0 and M_1; M_1 and M_2; M_2 and M_3 respectively. In next Step, i.e., in Step 3, we find the shortest path between $u = 2 = e_0$ and $v = 18 = e_4$. For this purpose, we find $e_{i-1} \hookrightarrow e_i$, i.e., the shortest path between e_{i-1} and e_i from the block M_{i-1}, for $i = 1, 2, 3, 4$ as follows:

$e_0(= 2) \hookrightarrow e_1(= 10)$ is $2 \mapsto 3 \mapsto 10$,
$e_1(= 10) \hookrightarrow e_2(= 16)$ is $10 \mapsto 11 \mapsto 16$,
$e_2(= 16) \hookrightarrow e_3(= 17)$ is $16 \mapsto 17$,
$e_3(= 17) \hookrightarrow e_4(= 18)$ is $17 \mapsto 18$.

In Step 4, we compute $d(2, 10) = 2 = r_0$, $d(10, 16) = 2 = r_1$, $d(16, 17) = 1 = r_2$, $d(17, 18) = 1 = r_3$. In Step 5, we compute $d(2, 18) = r_0 + r_1 + r_2 + r_3 = 2 + 2 + 1 + 1 = 6$, and the shortest path between 2 and 18 is $2 \hookrightarrow 10 \hookrightarrow 16 \hookrightarrow 17 \hookrightarrow 18$, i.e., $2 \mapsto 3 \mapsto 10 \mapsto 11 \mapsto 16 \mapsto 17 \mapsto 18$.

In Step 6, we set $i = 0$.

Step 7 and Step 8: At first, in Step 7, $l_0 = 4$ and $\frac{l_0}{2} = r_0$. So, we set $p_0 = M$ and $i = i + 1 = 1$. In Step 8, $i \neq 4\ (= l + 1)$, so, go to Step 7. In Step 7, $l_1 = 4$ and $\frac{l_1}{2} = r_1$. So, we set $p_1 = M$ and $i = i + 1 = 2$. In Step 8, $i \neq 4\ (= l + 1)$, so, go to Step 7. In Step 7, $l_2 = 4$ and $\frac{l_2}{2} = r_2$. So, we set $p_2 = l - 2r_2 = 4 - 2(1) = 2$ and $i = i + 1 = 3$. In Step 8, $i \neq 4 (= l + 1)$, so, go to Step 7. In Step 7, $l_3 = 2$. So, we set $p_3 = M$ and $i = i + 1 = 4$. In Step 8, $i = 4\ (= l + 1)$, so, we stop the Step 8 as well as Step 7.

After then, we compute $\min\{p_0 = M, p_1 = M, p_2 = 2, p_3 = M\} = p_2 = p_j = 2$ in Step 9. In Step 10, $d_n(2, 18) = d(2, 18) + p_2 = 6 + 2 = 8$ and next-to-shortest path between 2 and 18 is obtained by replacing $e_{j(=2)} \hookrightarrow e_{j+1}$, i.e., $e_2(= 16) \hookrightarrow e_3(= 17)$ by the next-to-shortest path between 16 and 17 (which is $16 \mapsto 19 \mapsto 21 \mapsto 17$) from $2 \hookrightarrow 18$. Hence, the required next-to-shortest path between 2 and 18 is $2 \mapsto 3 \mapsto 10 \mapsto 11 \mapsto 16 \mapsto 19 \mapsto 21 \mapsto 17 \mapsto 18$.

Theorem 1: *The time complexity to find next-to-shortest path between any two specified vertices u and v in a cactus graph G is $O(n)$, where n is the number of vertices of G.*

Proof. In Step 1, the BFS tree T can be constructed in $O(n)$ time as we know that $m = O(n)$ for cactus graphs. In BFS tree T, $d(B_i, B_j) = l$, so $l \leq n$. Therefore, in Step 2, the time complexity of marking the main path between the blocks B_i and B_j is $O(n)$. In Step 3, shortest path between e_{i-1} and e_i from M_{i-1}, for $i = 1, 2, \ldots, (l + 1)$ can be computed in $O(M_0 + M_1 + \cdots + M_l) = O(l_0 + l_1 + \cdots + l_l) = O(n)$ time as $m = O(n)$ for cactus graphs. Since, Step 4 and Step 5 are completed using the result of Step 3, so the run time of Step 4 and Step 5 is $O(n)$. Also Step 7 and Step 8 can be completed together in $O(n)$. In Step 9, $\min\{p_i,\ i = 0, 1, 2, \ldots, l\}$ can be computed on $O(n)$ time. Run time of Step 10 is also $O(n)$.

Hence, the overall time complexity of the **Algorithm CNSP** is $O(n)$.

6. CONCLUSION

Next-to-shortest paths problem has been widely studied in the past. In this paper, we proposed an $O(n)$ time algorithm to find next-to-shortest path between any two vertices of simple, connected and unweighted cactus graphs. We feel that it would be interesting to design an $O(n)$ time algorithm to find k shortest paths between two vertices of weighted cactus graphs.

ACKNOWLEDGEMENTS

We are thankful to our colleagues, scholars and friends for their continuous inspiration.

REFERENCES

[1] Barman, S. C., Mondal, S. and Pal, M. "An efficient algorithm to find next-to-shortest path on trapezoid graphs". Advances in Applied Mathematical Analysis, 2, pp. 97-107, 2007.

[2] Barman, S. C., Mondal, S. and Pal, M. "An efficient algorithm to find next-to-shortest path on permutation graphs". Journal of Applied Mathematics and Computing, 31, pp. 369-384, 2009.

[3] Chen, C. C. Y. and Das, S. K. ". Breadth- first traversal of trees and integer sorting". Parallel Information Processing Letters, 41**,** pp. 39-49, 1992.

[4] Das, K. and Pal, M. "An optimal algorithm to find a maximum weight 2-coloured set on cactus graphs". Journal of Information and Computing Science, 5 (3)**,** pp**.** 211-223, 2010.

[5] Das, K. and Pal, M. "An optimal algorithm to find minimum dominating set on cactus graphs". Mathematical Modelling and Applied Computing, 3 (1)**,** pp**.** 29-41, 2012.
[6] Das, K. and Pal, M. "An optimal algorithm to find a minimum 2-neighbourhood covering set on cactus graphs". Annals of Pure and Applied Mathematics, 2 (1)**,** pp. 45-59, 2012.
[7] Deo, N. "Graph theory with applications to engineering and computer science". (Prentice Hall of India Private Limited, New Delhi,), 1990.
[8] Eppstein, D. "Finding the k shortest paths*". SIAM J. Comput*, 28**,** pp. 652-673, 1998.
[9] Golumbic. M. C. ". Algorithmic graph theory and perfect graphs". (Academic Press, New York,), 1980.
[10] Khan, N., Pal, A. and Pal, M. "Edge colouring of cactus graphs". *Advanced Modelling and Optimization*, 11(4), pp. 407-421, 2009.
[11] Khan, N., Pal, A. and Pal, M. "(2, 1)-total labelling of cactus graphs". Journal of Information and Computing Science, 5(4), pp. 243-260, 2010.
[12] Khan, N., Pal, A. and Pal, M. "L(0, 1)-Labelling of cactus graphs". Information Processing Letters, 4, pp. 18-29, 2012.
[13] Khan, N., Pal, A. and Pal, M. ". Cordial labelling of cactus graphs" Advanced Modelling and Optimization, 15, pp. 85-101, 2013.
[14] Khan, N., Pal, A. and Pal, M. "Adjacent vertex distinguishing Edge colouring of cactus graphs". International Journal of Engineering and Innovative technology, 4(3), pp. 62-71, 2013.
[15] Koontz, W.L.G. "Economic evaluation of loop feeder relief alternatives". Bell System Technical J. 59, pp. 277-281, 1980.
[16] Krasikov, I. and Noble, S. D. "Finding next-to-shortest paths in a graph". Information Processing Letters, 92**,** pp. 117-119, 2004.
[17] Lalgudi, K. N., Papaefthymiou, M. C. and Potkonjak, M. "Optimizing systems for effective block-processing: The Kdelay problem". Tech. Rept. YALE/ DCS/ RR-1102, Dept. of Computer Science, (Yale University), 1996.
[18] Lalgudi, K. N. and Papaefthymiou, M. C. "Computing strictly-second shortest paths". Information Processing Letters, 63**,** pp. 177-181, 1997.
[19] Maimon, O. and Kincaid, R.K. "Material handling system design for flexible manufacturing systems: A network approach". Working Paper, Department of Mathematics, College of William and Mary, Williamsburg, VA.
[20] Maity, K., Pal, M. and Pal, T. K. "An optimal algorithm to find all pairs shortest paths problems on weighted cactus graphs". Journal of Physical Sciences, 6**,** pp. 45-57, 2000.
[21] Mandal, S. and Pal, M. "An optimal algorithm to solve next-to-shortest path problem on circular-arc graphs". Journal of Physical Sciences, 10, pp. 201-217, 2006.
[22] Mondal, S., Pal, M. and Pal, T. K. "An optimal algorithm to solve the all-pairs shortest paths problem on permutation graphs". Journal of Mathematical Modelling and Algorithms, 2, pp. 57-65, 2003.
[23] Mondal, S., Pal, M. and Pal, T. K. "An optimal algorithm for solving all-pairs shortest paths on trapezoid graphs". Intern. J. Comput. Engg. Sci., 3(2), pp. 103-116, 2002.
[24] Muller, J. H. and Spinrad, J. "Incremental modular decomposition". J. ACM, 36, pp. 1-19, 1989.
[25] Nishi, T. and Chua, Leon O. "Topological proof of the Nielsen-Willson theorem". IEEE Transactions on Circuits and Systems, 33 (4), pp. 398-405, 1986.
[26] Nishi, T. and Chua, Leon O. "Uniqueness of solution for nonlinear resistive circuits containing CCCSs or VCVSs whose controlling coefficients are finite". IEEE Transactions on Circuits and Systems, 33 (4)**,** pp. 381-397, 1986.
[27] Nishi, T. "On the number of solutions of a class of nonlinear resistive circuit". Proceedings of the IEEE International Symposium on Circuits and Systems, Singapore, pp. 766-769, 1991.
[28] Pal, M. "Intersection graphs: An introduction". Annals of Pure and Applied Mathematics, 4 (1), pp. 43-91, 2013.
[29] Paten, B., Diekhans, M., Earl, D., St. John, J., Ma, J., Suh, B. and Haussler, D. "Research in Computational Molecular Biology". Lecture Notes in Computer Science, 6044, *pp*. 410-425, 2010.
[30] Pnueli, A., Lempel. A. and Even, S. "Transitive orientation of graphs and identification of permutation graphs". Canad. J. Math., 23, pp. 160 – 175, 1971.
[31] Saha, A., Pal, M. and Pal, T. K. "An optimal parallel algorithm to find all-pairs shortest paths on circular-arc graphs". Journal of Applied Mathematics and Computing, 17 (1-2), pp. 1-23, 2005.
[32] Seidel, R. "On the all pairs shortest path problem". In Proceedings of 24th ACM Press, pp. 745-749, 1992.
[33] Li, S., Sun, G. and Chen, G. "Improved algorithm for finding next-to-shortest paths in a weighted undirected graph". Information Processing Letters, 99**,** pp. 192-194, 2006.
[34] Tarjan, R. E. "Depth first search and linear graph algorithm". SIAM J. Comput., 2**,** pp. 146-160, 1972.
[35] West, D. B. "Introduction to graph theory". Prentice Hall, 2**,** 2001.