Unit Rectangle Visibility Representation of Binary and Ternary Tree

Mehnuma Tabassum Omar, Kamrun Naher, Monika Gope and Ummay Asma

Department of Computer Science and Engineering
Khulna University of Engineering & Technology (KUET)
Khulna 9203, Bangladesh.

Abstract: The Visibility Representations [1] of a graph has been promoted for research extensively because of their significance in algorithmic graph theory as well as in VLSI layout, algorithm animation, visual languages and CASE tools etc. [2][3]. Rectangle Visibility Graph (RVG) used in VLSI chip design represents the node of a graph as a rectangle in a plane. Unit Rectangle Visibility Graph (URVG) means a presentation where each node is represented as unit rectangle in the plane. So, Unit Rectangle Visibility graph is applied for fixed dimensions of gates and various circuit components in computer chip applications. In this research, binary tree and ternary tree have been characterized as URV representations which will not only enhance URVR (Unit Rectangle Visibility Representation) but also expected to reduce both cost and labor in the field of various graph applications. To achieve this, two novel algorithms for URVR of these trees have been proposed.

Keywords: Visibility Representation, Unit Rectangle Visibility Graph, flush, caterpillar, spine.

I. INTRODUCTION

The representations of graph have been analyzed exclusively due to their intentions to be applied in numerous visibility applications. One of the representations is to employ geometric entity to depict a graph entity. This initiative has been inspired by the diversity of practical significances and the potentiality of interchange between geometry, graph theory and other mathematical area [4]. The Visibility Representation of graph is one of the natural strategies of representing graph using geometric aspects [5]. The determination of Visibility Representation (VR) of a graph has been studied widely because of their large number of applications such as in VLSI design, CASE tools, hidden surface elimination problem, layout problem [6].

Visibility Representation or VR demonstrates the vertices of a graph as horizontal vertex- segments, and the edge as vertical edge-segment. The edge maintains its end vertices by touching their corresponding segments without intersecting the segments beyond that. In before, for large number of applications and layout problems have motivated the study of Bar Visibility Graph (BVG) and Rectangle Visibility Graphs (RVGs) which is a two-dimensional counterpart of BVG [7].

Bar visibility graph (BVG) is one in which each vertex of a graph is determined by a horizontal bar and edges by unobstructed vertical visibilities between the corresponding bars. The adjacent vertices are presented by adjacent bars. The adjacent bars are depicted by a vertical band positive width that intersects no other bars [6]. A Unit Bar Visibility Graphs (UBVGs) is a BVG having bars of unit length.

In a Rectangle Visibility Graph (RVG) the vertices of a graph is resolved by closed rectangles in a plane and edges by unobstructed horizontal or vertical visibilities between the corresponding rectangles. The sides of the rectangles are parallel to the axes pair wise disjoints except possibly for overlapping boundaries, in such a way that two vertices are adjacent if and only if each of the corresponding rectangles is vertically or horizontally visible from the other [7]. The BVG and RVG are applicable in VLSI chip design and layout problems.

A Unit Rectangle Visibility Graph (URVG) is a RVG having rectangles of unit squares. In BVG and RVG the extent of bars and rectangles respectively may have diverse dimensions. But there are some constraints in the area and aspect ratio for the component of a chip [8]. The URVG maintains unit squares in its rectangles and for this reason it preserves a fixed area and fixed aspect ratio. So, in chip design, URVG achieves popularity than the other representations. URVG converts a graph into a model that is more precise to the subject [7]. It is also efficient in designing computer chips having huge number of logical gates with complicated wiring and timing constraints.

This paper describes the Unit Rectangle Visibility Trees (URVR) of a graph where each vertex can take place and satisfy all of the Unit Rectangle Visibility Representation conditions. The presentation is dedicated explicitly on binary tree and ternary tree and developed two algorithms for these trees.

The paper is organized in six sections. In section II the URVG representation is described briefly. Section III will discuss related works so far been performed for the presentation of graph. The proposed methodologies are described in section IV. Finally section V will conclude this paper.

II. THE UNIT RECTANGLE VISIBILITY GRAPH

The function of RVG is to transform nodes of a graph to some rectangles. URVG is an extension of RVG. It restricts the rectangles to be arranged in unit squares so that it can form more subjective applications.

Fig. 1 shows an example of a graph with its URV layout. The square in the URV layout corresponding to a vertex v is symbolized by S_v [7]. The placement of the square S_v in a

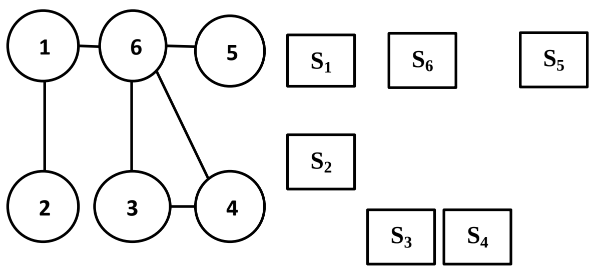

Figure 1: A graph with a URV layout

URV layout is performed by the bottom-left corner coordinates (x_v, y_v) of the URV layout. In this layout, X_v is a line segment given by the intersection of the line $x = x_v$ with the square S_v, and Y_v is a line segment given by the intersection of the line $y = y_v$ with the square S_v. The S_v and S_w will be entitled as a flush if the co-ordinate x_v is equal to the co-ordinate x_w or the co-ordinate y_v is equal to the co-ordinate y_w. But flush does not prohibit the other squares in the layout to hamper the visibility between S_v and S_w. A flush will be defined as a horizontal flush if the coordinates X_{Sv} and X_{Sw} are equal. A vertical flush is a flush where the coordinates Y_{Sv} and Y_{Sw} are equal. In the given layout depict in Fig. 1, the squares S_2 and S_3 are collinear but not flush while the squares S_1 and S_2 and the squares S_1, S_5, and S_6 are flushes.

RELATED WORKS

The VR representation of a graph is a geometrical presentation in the arena of graph which is very popular of being relevant in the implementation of VLSI layout [9]. The significance of VR is so fascinating that it keeps recognition of several researchers [9] [10]. This section will refer to some of the variants of VR viewing different significant applications.

In [11] they claimed a VR as an augmentation of tessellation representation. According to this consequence the general dilemma with most implementations of the various tessellation representations are robustness as it identifies the surfaces as edges. If topology information is overlooked during the tessellation method, nearby surfaces may have contrary borders and thus do not stick together easily. In [1] to present vertices of a graph, a group of parallel hyper-rectangles in R^n is provided with the visibility orthogonal to the hyper-rectangles. The non-planner bar k-visibility representation of graph is demonstrated in [12]. [17] Illustrated the area specification of bar visibility representation and rectangle visibility of trees in a plane. The survey of Rectangle Visibility Graphs (RVGs) is very potential in graph-visualization and graph-drawing domains [13]. An extension of RVG is demonstrated in [8] where the RVG presents each vertex of a graph by the rectangles having length of unit squares. The BVG and RVG are failed to meet the requirement of fixed area and fixed size aspect ratio of a computer chip because of their dimension diversity. For this concern [8] deliberated Unit Bar Visibility Graphs (UBVGs) having bars of identical length. A variant of this scheme that utilizes 3-space boxes was considered in [14], [15]. The RVG representation can also be used in tree applications. In [16] the prerequisite of a k-tree to be RVG are narrated. If the range for k is $1 \leq k \leq 4$ then the tree became a RVG. In [18] a definition for 1-tree to be a URVG is presented. The paper also summarized that a 4-tree is not a URVG. [19] Emphasizes the ternary URVG presentation. They disclosed

Input : Binary tree

Output : URVR presentation of binary tree

Algorithm :

```
URV_binary_tree( ){
        //enter node info
  1. Set position for root at $x_i$,$y_i$.
  2. Find root of current node
        //check node , left or right child of root
  3. if(node==left child)
            //check level, odd or even
            if(level==odd)
                set position at min_x-2,$y_i$-2/3
            if(level==even)
                set position at $x_i$-2/3,min_y-2
            find max-min position
  4. if(node==right child)
            //check level, odd or even
            if(level==odd)
                set position at max_x+2,$y_i$+2/3
            if(level==even)
                set position at $x_i$+2/3,max_y+2
            find max-min position
  5. increase node value by one
  6. Repeat step 2-5 until node remain
  7. Draw rectangle using position of nodes
  8. Draw node number with rectangles.
}
```

Figure 2: Algorithm for URVG representation of a binary tree

a little about the binary presentation.

The paper showed that if the decomposition of ternary tree has maximum 4 degree and a union of three caterpillars with additional properties then the presentation is a URVG. A tree where the degree of all vertices on a single path is greater than 1 is named as a caterpillar. If the length of the path is maximal then it is defined as a spine of the caterpillar. But this presentation is not sufficient. In this paper two unique algorithms have been suggested for binary and ternary tree which will enrich the URVR presentation.

PROPOSED METHODOLOGY

A tree is an undirected graph in which any two vertices are connected by exactly one simple path. In other words, any connected graph without cycles is a tree. A forest is a disjoint union of trees.

A binary tree is a tree data structure in which, each node has at most two child nodes, usually distinguished as "left" and "right". A tree data structure which has at most three child nodes is called a ternary tree. A node is a structure which may contain a value, a condition, or represent a separate data structure (which could be a tree of its own). Each node in a tree has zero or more child nodes, which are below it in the tree (by convention, trees are drawn growing downwards). A node that has a child is called the child's parent node. A node has at most one parent. An internal node or inner node is any node of a tree that has child nodes. Similarly, an external node (also known as an outer node, leaf node, or terminal node), is any node that does not have child nodes. The level of a node is defined by the number of connections in the shortest path between node and the root [20].

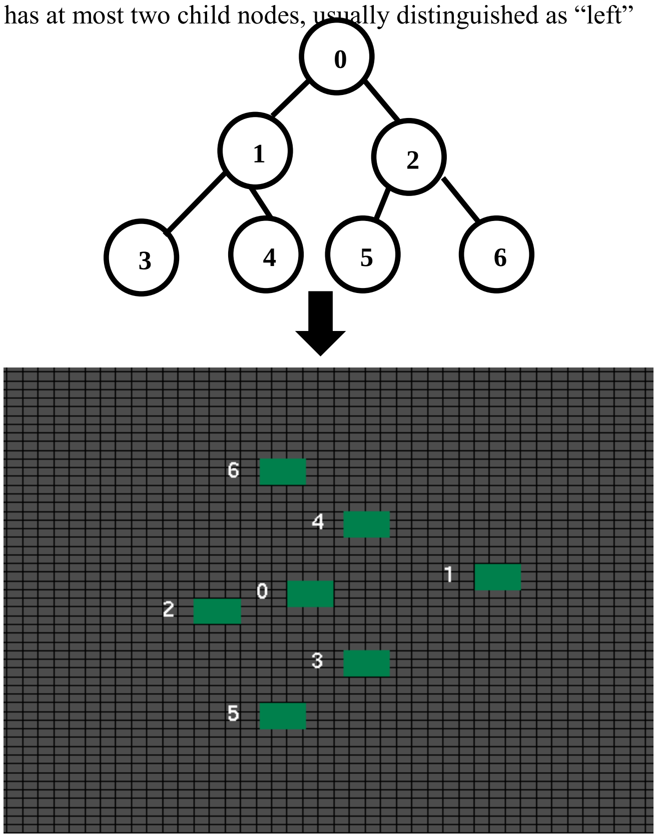

Figure 3: Input and output of URVR binary presentation

In this paper two algorithms for URVR of trees have been proposed. The first algorithm works for binary trees and the later one will work for ternary trees. For presenting a tree on URVR, some co-ordinate positions will be required. The root of the tree will be traced by the co-ordinate position (x_i, y_i). Some other positions like x_m, x_M, y_m, y_M will be maintained for minimum x coordinate, maximum x coordinate, minimum y coordinate and maximum y coordinate respectively. The levels of the tree like level0, level1, level2 will be declared by L0, L1and L2.

A. URV Representation of Binary Tree

This section will convert a binary tree to a URVR. This binary tree may be balanced or unbalanced. The proposed algorithm will work for both the balanced and unbalanced tree. First the root will be placed. Then the left child and the right child will be placed accordingly. The algorithm is illustrated in Fig. 2.

The following steps will execute the proposed approach for converting a binary tree to a URVR.

Input : Ternary tree

Output : URVR presentation of ternary tree

Algorithm :

```
URV_ternary_tree( ){
    1. draw root [0] at initial co-ordinate (x,y)
    2. define some points as
       D1 = ($X_M$+3, $Y_i$+1/4)
       D2 = ($X_m$ -3, $Y_i$ -1/4)
       D3 = ($X_i$+1/4, $Y_M$+3)
       D4 = ($X_i$ -1/4, $Y_m$ -3)

       And some pattern as
       P1 = D3, D4, D1
       P2 = D3, D4, D2
       P3 = D1, D2, D3
       P4 = D1, D2, D4

    3. draw level-1 nodes 1,2,3 at points D1, D2, D3
       considering root=[0] means i=0;
    4. increment level means L=2 and set i=1 and draw
       child of i to looking at its root point
       If it is D1 then draw pattern P1
         Else if it is D2 then draw pattern P2
           Else if it is D3 then draw pattern P3
             Else if it is D4 then draw pattern P4
    5. increment i and repeat step 4 two more times as
       each node have 3 child
    6. increment level and go to step 4 for drawing next
       level's child node
    7. repeat step 6 until it reaches the last level
}
```

Figure 4: Algorithm for URVG representation of a ternary tree

a) Placing Root : For depicting a URVR from binary tree, at first the root is placed. Each node is presented by a rectangle which is unit in size. For placing a unit edge the length is maintained eqaul and the distance is maintained as one unit. The bottom left corner of a rectangle is searched. Then the rectangle is placed the at any arbitrary position, x_i, y_i.

b) Placing Left Child, odd level **:** Start with left child. As root is the first node, it is on L0. So its child remains at L1. Then the child of L1is placed at L2 which is an even level. So, as child of root is at L1, that is in odd level, it is placed at position (x_m-2, y_i-2/3). The value of x_m, x_M, y_m, y_M is updated every time after placing each rectangle.

c) Placing right child, odd level : Start with right child. It is placed at position (x_i-2/3, y_m-2). After placing child it is necessary to check whether it overlaps with other or not.

d) Placing left child, even level : Starting with L2, that is, even level. So according to the even level rule, the left node of child one is placed at position (x_M+2,y_i+2/3).

e) Placing right child, even level : Place right child at (x_i+2/3, y_M+2). That is rule for even level right child.

Example 1: The start node is selected as root node and placed at any arbitrary location in URVG layout. The root node is represented at L0 and symbolized as node 0.

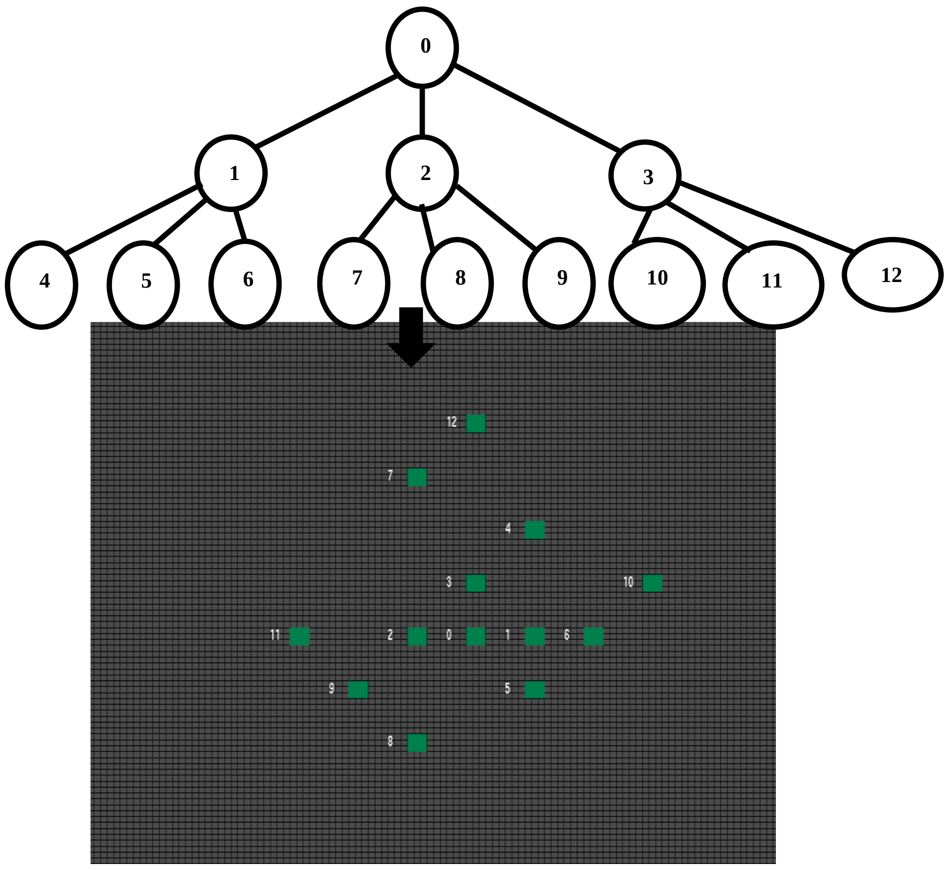

Figure 5: Input and output of URVR ternary presentation

From root node two child nodes are produced entitling as left child and right child. These child nodes are symbolized as node 1 and 2. Node 1 is placed at position (XM+2, Yi+2/3) and 2 is placed at position (X_m-2, Y_i-2/3). Update the coordinate value by incrementing the value for i. From L1 another level known as even level L2 is created. Node 1 produces two child denoted by node 3 and 4 are placed into the URV layout using the even level process. The co-ordinates selected for left child is (X_i-2/3, Y_m-2) and for right child is (X_i+2/3, Y_M+2). Node 2 produces two child node denoted by 5 and 6. The co-ordinates selected for left child or node 5 is (X_i-2/3, Y_m-2) and for right child or node 6is (X_i+2/3, Y_M+2).

So every level follows this condition properly and without overlapping the square is placed successfully. The above example including input and corresponding output is depicted in Fig. 3

B. URV Representation of Ternary Tree

This section will convert a terary tree to a URVR. The idea is to place the root will first and then the left child and the right child will be placed accordingly. The algorithm is illustrated in Fig. 4. This terary tree is restricted to be a balanced tree. The following steps will execute the proposed approach for converting a ternary tree to a URVR.

a) Placing root : Each node is presented by a rectangle which is unit in size. For placing a unit edge the length is maintained eqaul and the distance is maintained as one unit. The bottom left corner of a rectangle is searched. Then the rectangle is placed the at any arbitrary position, x_i, y_i . Now define some points as D1 = (X_M+3, Y_i+1/4), D2 = (X_m -3, Y_i -1/4), D3 = (X_i+1/4, Y_M+3), D4 = (X_i -1/4, Y_m -3) and some patterns as P1 = D3, D4, D1; P2 = D3, D4, D2; P3 = D1, D2, D3; P4 = D1, D2, D4.

b) Placing childs at L1 : Start with the left child. As root is the first node, it is on L0. So its child remain at L1. The child of L0 is placed at position D1, D2, D3. The first child is placed at position D1 that is at (X_M+3, Y_i+1/4), second child at position D2 that is (X_m -3, Y_i -1/4) and finally the third child at position D3 that is at (X_i+1/4, Y_M+3).

c) Placing childs at L2 : Then child of L1 is placed at L2 that is at even level. As child of root is at L1, that is in odd level, for L2 the position of the parent of L2 is used. If the parent is at D1, then the pattern P1 is used for its three child. That is D3, D4 and D1. If the parent is at position D2, then use pattern P2 that is D3, D4, and D2. If the parent is at position D3, then use pattern P3 that is D1, D2, D3. The value of X_m, X_M, Y_m, Y_M is updated every time after placing each rectangle.

d) Placing childs at L3 : At L3 another pattern P4 that is D1, D2, D4 is used because the parent is now at L2 and it can be at position D4 that is X_i -1/4, Y_m -3. Another patterns are also be used here.

Thus all children at both odd and even level are placed according to the given rules. Here the steps are defined up to L3. The above example including input and corresponding output is depicted in Fig. 5.

CONCLUSIONS

Chip designing is the key part of chip design model in VLSI design or in fabrication. Visibility Representation is a drawing tool to be used in this purpose. Unit Rectangle Visibility Graph (URVG) arranges each node of a graph as unit rectangle in the plane and gives fixed area and aspect ratio while designing a chip. So Unit Rectangle Visibility Representation is significant in this aspect. The determination of this paper is to define an efficient algorithm for an arbitrary graph which will increase the performance of UVG and decrease the cost and labor while designing. For this purpose, we have offered two new algorithms, named as URV_ binary_tree and URV_ternary_tree. We have represented the binary tree and ternary tree in a URVR and have accomplished our proposed algorithms successfully. If the graph is large enough, complex, then connection is overlapped to another node. But the existing algorithm works on trees having restriction of degree three. Our proposed algorithm, for ternary tree is implemented for maximum degree four. These proposed algorithms will give a good direction in URV representation.

REFERENCES

[1] F.J. Cobos, J.C. Dana, F. Hurtado, A. M´arquez and F. Mateos,"On a Visibility Representation of Graphs". volume 1027 of Lecture Notes in Computer Science.Springer- Verlag, 1995.
[2] Huaming Zhang and Xin He, "Improved visibility representation of plane graphs", Computational Geometry 30 (2005) 29–39.
[3] Ioannis G. Tollis, "Visibility representations of planar graphs (abstract)", ACM SIGACT News Homepage archive Volume 24 Issue 1, Winter 1993Pages 57-58.
[4] Stefan Felsner," Rectangle and Square Representations of Planar Graphs", Partially supported by DFG grant FE-340/7-2 and the EUROGIGA project GraDR.
[5] Alice M. Deana and Joan P. Hutchinsonb, "Rectangle-visibility representations of bipartite graphs", Discrete Applied Mathematics 75 (1997) 9-25.
[6] Jiao-Hao Fan, Chun-Cheng Lee, Hsueh-I Lu, Hsu-Chun Yen, "Width-Optimal Visibility Representations of a Plane Graph", Algorithms and Computation: 18th International Symposium, ISAAC, 2007.
[7] Alice M. Dean, Joanna A. Ellis-Monaghan, Sarah Hamilton, Greta Pangborn, "Unit Rectangle Visibility Graphs", Mathematics Subject Classification: 05C62, 2008.
[8] A. M. Dean and N. Veytsel. Unit bar-visibility graphs. Congressus Numerantium, 160:161–175, 2003.
[9] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. Graph Drawing: Algorithms for the Visualization of Graphs. Prentice-Hall, Upper Saddle River, New Jersey, 1999.
[10] F. J. Cobos, J. C. Dana, F. Hurtado, A.Marquez, and F. Mateos.On a Visibility Representation of Graphs, volume 1027 of Lecture Notes in Computer Science.Springer- Verlag, 1995.
[11] Ákos Balázs dissertation of the doctoral degree Tessellation and rendering of trimmed NURBS models in scene graph systems. University of Bonn, (2008).
[12] M. Dean, W. Evans, E. Gethner, J. D. Laison, and M. Safari. Bar k-visibility graphs. Journal of Graph Algorithms and Applications, 11(1):45–59, 2007.
[13] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. Graph Drawing. Prentice Hall, Upper Saddle River, NJ, 1999.
[14] P. Bose, A. Josefczyk, J. Miller, and J. O'Rourke. k42 is a box visibility graph. Technical Report #034, Smith College, 1994.
[15] S. P. Fekete and H. Meijer. Rectangle and box visibility graphs in 3d. International Journal of Computational Geometry and Applications, 9(1):1–21, 1999.
[16] P. Bose, A. Dean, J. Hutchinson, and T. Shermer. On rectangle visibility graphs. In Lecture Notes in Computer Science 1190: Graph Drawing, pages 25–44. Springer-Verlag, 1997.
[17] G. Kant, G. Liotta, R. Tamassia and I. G. Tollis, "Area requirement of Visibility Representations of Trees". In Proc. 5th CCCG, pages 192-197 ,Waterloo, Canada, 1993,
[18] A. M. Dean, J. A. Ellis-Monaghan, S. Hamilton, and G. Pangborn. Unit rectangle visibility graphs. Electronic Journal of Combinatorics, 15(#R79):1–24, 2008.
[19] Alice M. Deana and Joan P. Hutchinsonb, "Representing 3-trees as Unit Rectangle-Visibility Graphs", Congressus Numerantium, 2010.
[20] Alexander Kuznetso, Igor Pak, Alexander Postnikov, "Trees Associated with Motzkin Numbers". Journal of Combinatorial Theory, Series A 76, 145-147(1996), Article No. 0094.

AUTHORS

First Author – Mehnuma Tabassum Omar, B.Sc(CSE), M.Sc(CSE on going), Khulna University of Engineering & Technology, misty2409@gmail.com

Second Author – Kamrun Naher, B.Sc(CSE), Khulna University of Engineering & Technology

Third Author – Monika Gope B.Sc(CSE), M.Sc(CSE on going), Khulna University of Engineering & Technology, gopenath14@yahoo.com

Fouth Author – Ummay Asma, B.Sc(CSE), Khulna University of Engineering & Technology