Analysis and Visualization of Prim's Algorithm

Anand Kumar
Information Technology
NITK, Surathkal
anand.191it104@nitk.edu.in

Ashok Kumar
Information Technology
NITK, Surathkal
aashokkumar.191it210@nitk.edu.in

Vishal Kumar Verma
Information Technology
NITK, Surathkal
vishal.191it258@nitk.edu.in

***Abstract*—The spanning tree of a graph is defined as a subgraph that is a tree which includes all of the vertices of graph. In general, a graph may have several spanning trees, but a graph that is not connected will not contain a spanning tree. A minimum spanning tree (MST) or minimum weight spanning tree is a subset of the edges of a connected, edge-weighted undirected graph that connects all the vertices together, without any cycles and with the minimum possible total edge weight.**

There are different algorithms that are used to find the Minimum Spanning Tree of a graph i.e. Prim's Algorithm, Kruskal's algorithm etc. The main objective of our project is to discuss,analyse and make a visulalizer for the formation of Minimum Spanning Tree using Prim's Algorithm and for make the visualizer we have used the OpenGL graphic library of Cpp.

***Index Terms*—Minimum Spanning Tree (MST), Prim's Algorithm, Dense graph,Sparse Graph,OpenGL.**

I. Introduction

From graph perspective, a Tree is a subset of a directed or undirected connected graph in which vertices are linked with exactly one path without any cycles. A spanning tree of a connected graph can be constructed including all the vertices with minimum possible no of edges. If there are n vertices in the graph, then each spanning tree has n-1 edges. A connected weighted graph where all the vertices are interlinked by some weighted edges can contain multiple numbers of spanning trees. A spanning tree with minimum possible total edge weight is called as a Minimum Spanning Tree (MST). In the minimum spanning tree a set of edges is selected such that there is a path between each node and the sum of the edge weights is minimal. A minimum spanning tree can be constructed using greedy algorithms i.e. Prim's and Krushkal algorithm. These algorithms are considered as greedy algorithm as they find the best smallest weight edges to build a MST. The kruskal's algorithm firstly processes the edges according to their weights from lowest to largest value. It takes the lowest valued edge and added to the MST. This process continues until all the vertices are visited. The Prim's algorithm is based on graph traversals. It first selects a random node from the graph and in each traversal, examines all edges from the visited node to non-visited nodes to select a minimum edge and then the edge is added to the MST.

II. System Specification

Software Requirements

Operating system	:	Ubuntu 18.x or above
Compiler used	:	G++
Programming language	:	C++ language
Editor	:	VScode
Graphics library	:	GL and GLU / GLUT

Hardware Requirements

Processor	:	Intel I3/I5/I7
Processor speed	:	1 GHz or more
Hard disk	:	40 GB or more
RAM size	:	1 GB or more
Display	:	800x600 or higher resolution display with 256 colours
Mouse	:	Standard serial mouse
Keyboard	:	Standard QWERTY keyboard
GPU	:	Intel HD Graphics 5000 or better

III. Prim's Algorithm

It is a greedy algorithm. It starts with an empty spanning tree. The idea is to maintain two sets of vertices: o Contain vertices already included in MST. o Contain vertices not yet included. At every step, it considers all the edges and picks the minimum weight edge. After picking the edge, it moves the other endpoint of edge to set containing MST.

Properties of Prim's Algorithm:

1) The edges in the subset of some minimum spanning tree always form a single tree.

2) It grows the tree until it spans all the vertices of the graph.

3) An edge is added to the tree, at every step, that crosses a cut if its weight is the minimum of any edge.

Working of Prim's Algorithm:

Steps for finding MST using Prim's Algorithm:

1. Create MST set that keeps track of vertices already included in MST.

2. Assign key values to all vertices in the input graph. Initialize all key values as INFINITE (). Assign key values like 0 for the first vertex so that it is picked first.

3. While MST set doesn't include all vertices.

a. Pick vertex u which is not is MST set and has minimum key value. Include 'u'to MST set.

b. Update the key value of all adjacent vertices of u. To update, iterate through all adjacent vertices. For every adjacent vertex v, if the weight of edge u.v less than the previous key value of v, update key value as a weight of u.v. Using the steps mentioned above, the working of the prim's algorithm is shown below using an example.

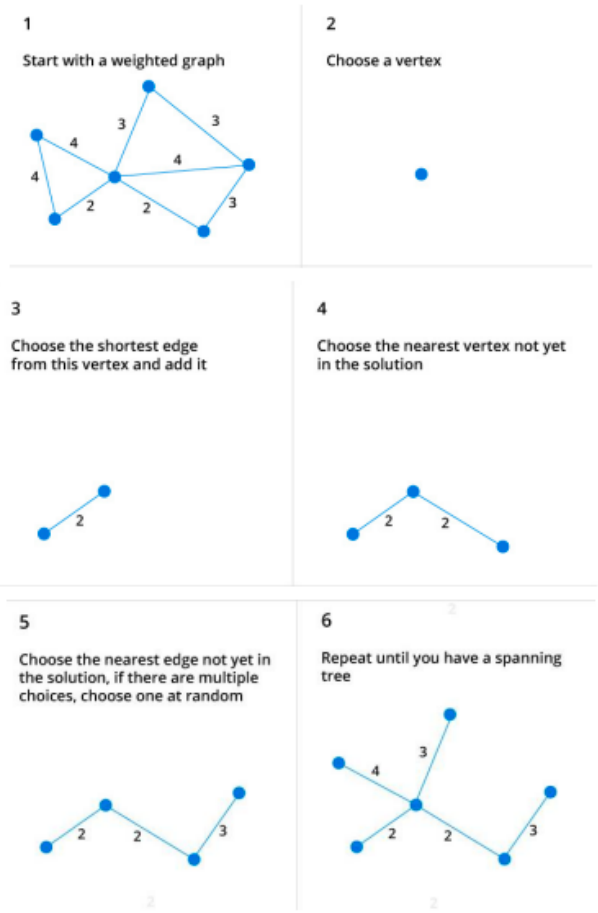

Fig. 1. Steps of constructing Prim's MST

IV. OpenGL

OpenGL is a software interface to graphics hardware. This interface consists of about 150 distinct commands that are used to specify the objects and operations needed to produce interactive three-dimensional applications. OpenGL is designed as a streamlined hardwareindependent interface to be implemented on many different hardware platforms. These are certain characteristics of OpenGL:

• OpenGL is a better documented API.
• OpenGL is much easier to learn and program.
• OpenGL has the best demonstrated 3D performance for any API.

The OpenGL specification describes an abstract API for drawing 2D and 3D graphics. Although it's possible for the API to be implemented entirely in software, it's designed to be implemented mostly or entirely in hardware.

Libraries present in OpenGL are shown in the figure given below :

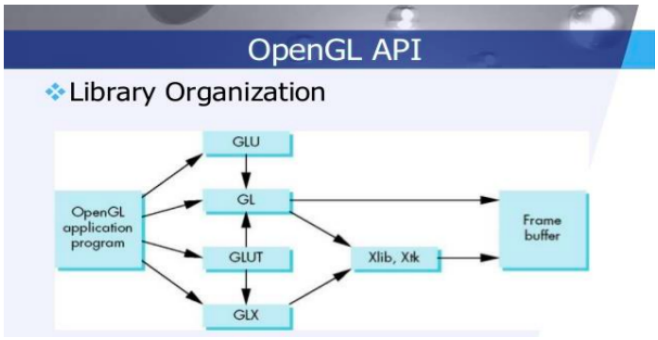

Fig. 2. OpenGL libraries

A. *Application of OpenGL*

• OpenGL (Open Graphics Library) is a cross-language, multi-platform API for rendering 2D and 3D computer graphics.
• The API is typically used to interact with a GPU, to achieve hardware accelerated rendering.
• It is widely used in CAD, virtual reality, scientific visualization, information visualization and flight simulation.
• It is also widely used in the development of video games for different platforms such as PC , consoles or smartphones

V. Design and Implementation

Introduction

Systems design is the process or art of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. One could see it as the application of systems theory to product development.

Initialization

Initialize the interaction with the windows. Initialize the display mode- double buffer and depth buffer. Initialize the various call-back functions for drawing and redrawing, for mouse and keyboard interfaces. Initialize the input and calculate functions for various mathematical calculations. Initialize the window position and size and create the window to display the output.

Flow Control

The flow of control in the below flow chart is with respect to the Texture Package. For any of the program flow chart is compulsory to understand the program. We consider the flow chart for the texture project in which the flow starts from start and proceeds to the main function after which it comes to the

initialization of call back functions and further it proceeds to mouse and keyboard functions, input and calculation functions. Finally, it comes to quit, the end of flow chart

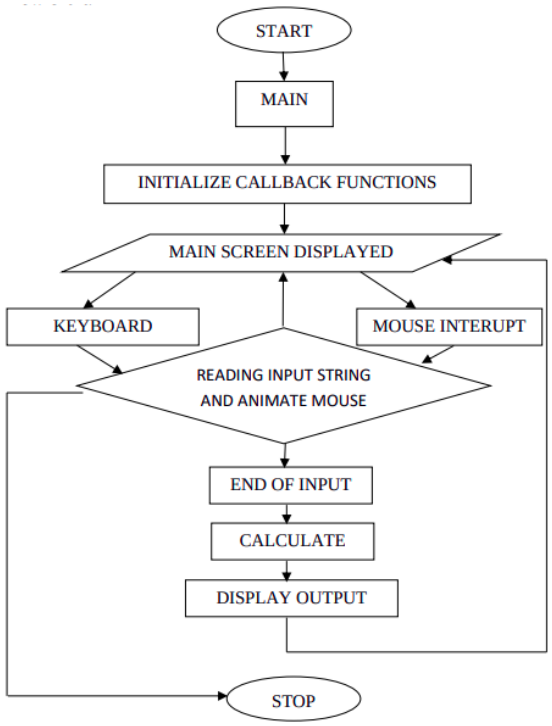

Fig. 3. Project Design

OpenGL APIs used/Built-in functions

• **glutPostRedisplay()** : Marks the current window as needing to be redisplayed.

• **glutBitmapCharacter()** : Renders a bitmap character using OpenGL from the specified array of characters, and in the specified font style.

• **glutTimerFunc()** : Registers a timer callback to be triggered in a specified number of milliseconds.

• **glClearColor()** : Specifies clear values for the color buffers.

• **glMatrixMode()** : Specifies which matrix is the current matrix.

• **glLoadIdentity()** : Pushes the identity matrix to the top of the matrix stack.

• **glutSwapBuffers()** : Swaps the buffers of the current window if double buffered.

• **glViewport()** : Sets the viewport.

• **glutInitDisplayMode()** : Sets the initial display mode.

• **glutInitWindowSize() and glutInitWindowPosition()** : Set the initial window size and position respectively.

• **glutCreateWindow()** : Creates a top level window with the window name as specified.

• **glutAddMenuEntry()** : Adds a menu entry to the bottom of the current menu.

• **glutAttachMenu()** : Attaches a mouse button for the current window to the identifier of the current menu.

• **glutDisplayFunc()** : Sets the display callback for the current window.

• **glutReshapeFunc()** : Sets the reshape callback for the current window.

• **glutMainLoop()** : Enters the GLUT event processing loop. This routine should be called at most once in a GLUT program. Once called, this routine will never return. It will call as necessary any callbacks that have been registered.

Pseudo code for algorithm

Prim's Algorithm to calculate Minimum Spanning Tree:

```
Prims()
{
    S = new empty set

    for i = 1 to n
        d[i] = infinity

    while S.size() < n
        x = infinity
        v = -1
        // V is the set of vertices
        for each i in V - S
            if x >= d[v]
                then x = d[v], v = i

        d[v] = 0
        S.insert(v)
        for each u in adj[v]
                do d[u] = min(d[u], w(v,u))

}
```

VI. Complexity

Finding the minimum distance is O (V) and overall complexity with adjacency list representation is O (V2).

If queue is kept as a binary heap, relax will need a decrease-key operation which is O (logV) and the overall complexity is O (V log V + E log V) i.e. O (E log V).

If queue is kept as a Fibonacci heap, decrease-key has an amortized complexity O (1) and the overall complexity is O (E+V log V).

VII. Results and Output

The project is executed using C++ language. We have put in few screen shots to show the working of Prim's Minimum Spanning Tree Algorithm.

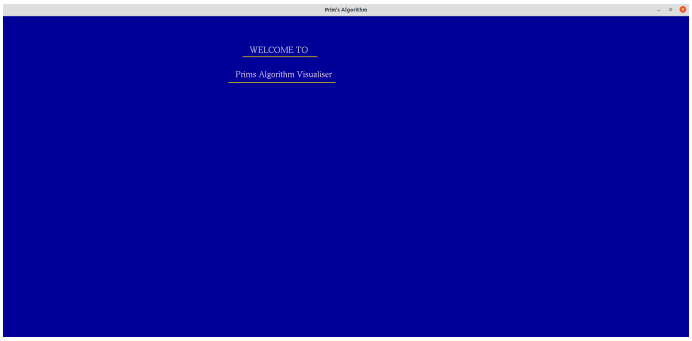

Fig. 4. Home Screen

Figure shows the home page of the program. Pressing 'Enter' on the keyboard, the user can move on to the main screen of the program.

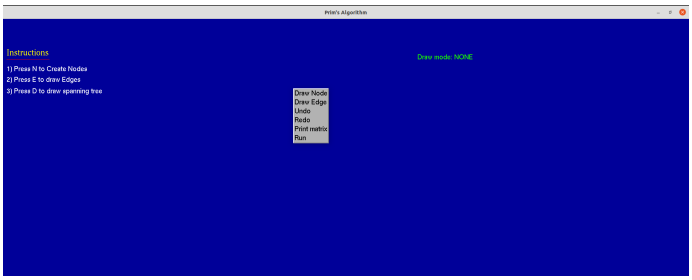

Fig. 5. Main screen

Figure shows the main screen of the program. From here the user can navigate to various modes of the program by pressing the key specified.

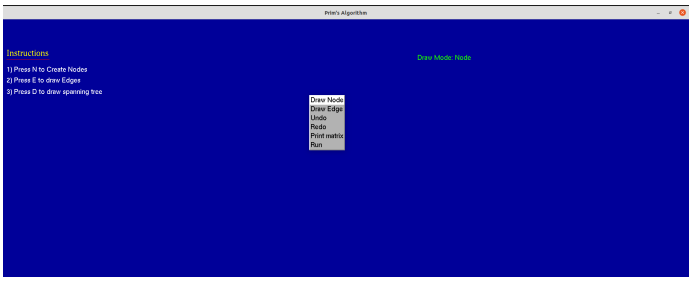

Fig. 6. Node Mode to create Nodes

Here the user can click anywhere in the window to create a node. It will display the node number when it is created.

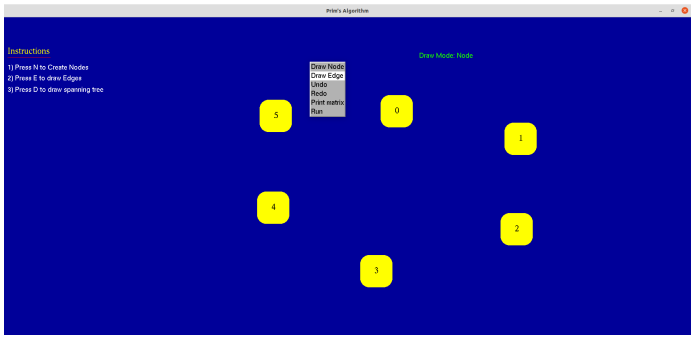

Fig. 7. All the nodes have been created. Press E to go to Edge Mode

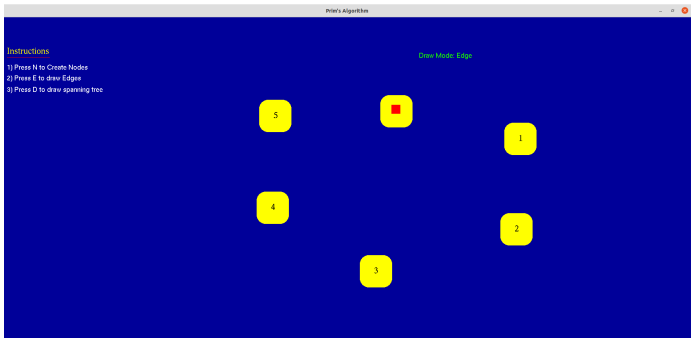

Fig. 8. The user can select the nodes to draw edge between them.

Figure shows the Edge mode in which the user is allowed to draw edges between the created nodes by selecting the starting and ending node.

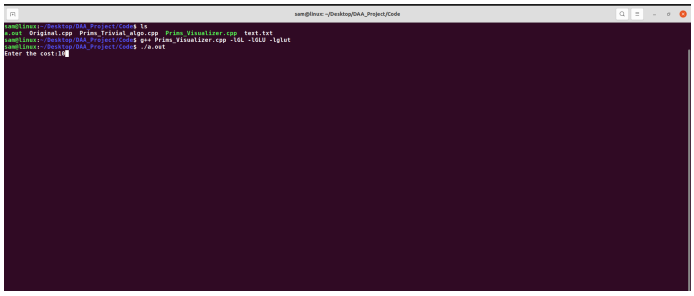

Fig. 9. Taking edge weight

After selecting the two edges the user has to provide the weight to the specified edge in order to draw it.

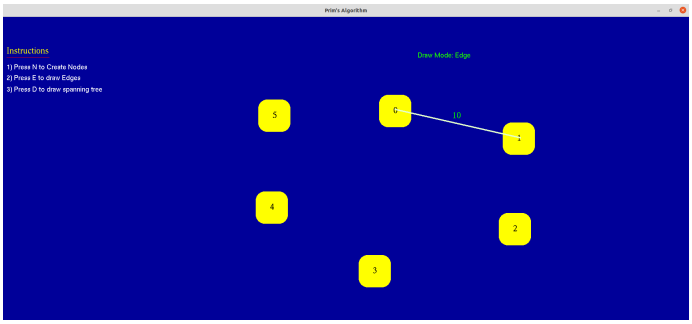

Fig. 10. Edges drawn between vertices with their costs

After taking the input of weight edge will appear in the graph as shown.

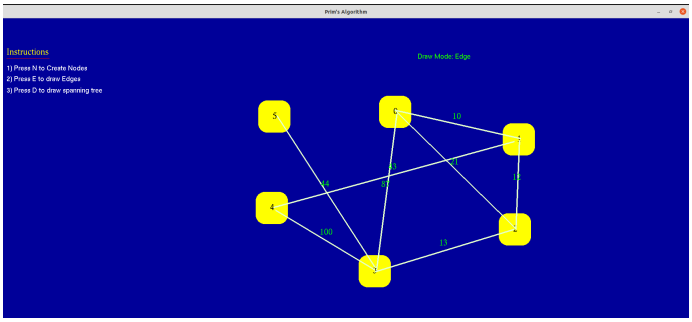

Fig. 11. All Edges are drawn to form connected graph.

Once we are done with drawing all the edges we are ready to go to the next step.

We have some more functionality in the application ie Undo, Redo your actions.

Choosing the print matrix option will give the output of the formed matrix of the graph.

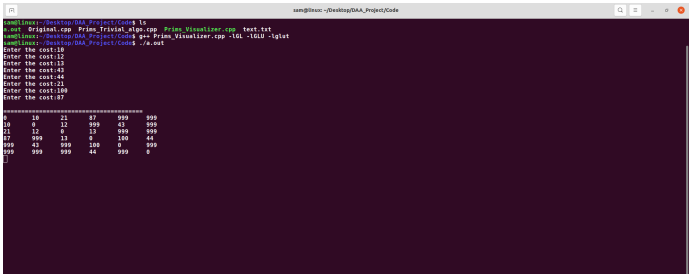

Fig. 12. Printing the matrix representation of formed graph

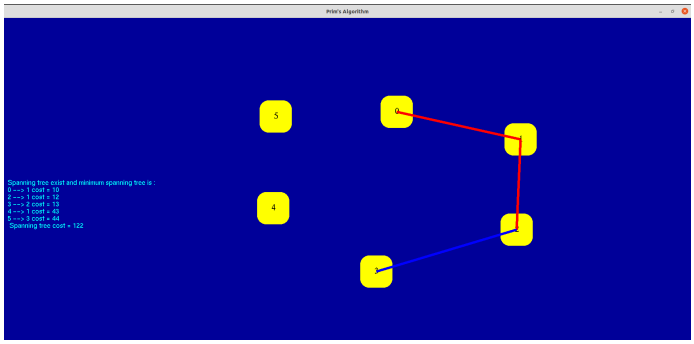

Fig. 13. After pressing D. The program calculates shortest path.

Once the user clicks D the finding the minimum spanning tree function would get called and corresponding path taken will be displayed as animation to the user with all the stats as shown in the figure.

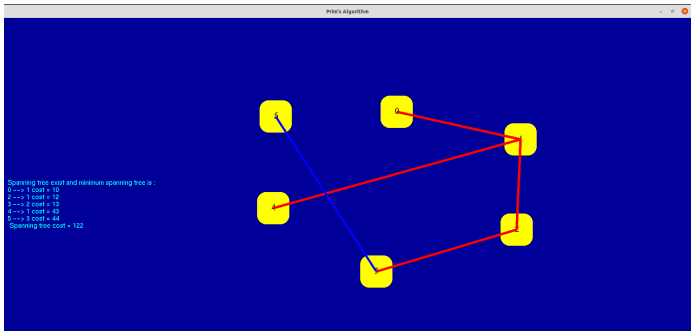

Fig. 14. Final animated Visualisation

It show the result or the shortest path by highlighting it between the nodes. The costs are also displayed. This can only be done if the graph is fully connected and when the user presses the D key on the keyboard.

Once the user is satisfied with the result, they may exit the window by pressing escape key on the keyboard.

VIII. Conclusion

The time complexity for Prim's algorithm is acceptable in terms of its overall performance to finding minimum spanning tree. Minimum Spanning Tree concept is mostly used in network analysis, water supply network, transportation network, cluster analysis, circuit design and to solve the travelling salesman problem of the graph theory. The Prim's algorithm is preferred to dense graphs but for sparse graph the Kruskal's algorithm is used for better results.

This mini project on Prim's Minimum Spanning Tree Algorithm using OpenGL is a reliable graphics package that provides a basic understanding of the working of the algorithm. It provides a visual representation of the algorithm with a user-friendly interface that allows the user to interact with it very effectively.

IX. Future Enhancements

The future scope of this project is quite large. This project can be further modified to take inputs in the form of matrices for more complicated graph construction to evaluate higher level problems. The current package can also be used to implement other similarly structured minimum spanning tree algorithms such as Kruskal's and Dijkstra's algorithms. The simple visuals can be enhanced with time to include further nuances and more use cases depending on the situation.

References

[1] H. J. Greenberg, Greedy Algorithms for Minimum Spanning Tree. Denver, 1998.

[2] Prim's Algorithm

[3] MST Greedy algorithms

[4] Computer Graphics using OpenGL

[5] OpenGL video tutorials

[6] G. Eason, B. Noble, and I. N. Sneddon, "COMPUTATIONAL METHODS FOR MINIMUM SPANNING TREE ALGORITHMS," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.