**D E L H I T E C H N O L O G I C A L U N I V E R S I T Y**



**DISCRETE STRUCTURES**

PROJECT REPORT

GRAPH GUI APPLICATION

**KHUSHI - 2K19/IT/067**

**HIMANSHI GUPTA - 2K19/IT/060**

**DELHI TECHNOLOGICAL UNIVERSITY**

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

**CANDIDATE’S DECLARATION**

We, **Khushi 2K19/IT/067 and Himanshi Gupta 2K19/IT/060**, students of B.Tech. (INFORMATION TECHNOLOGY), hereby declare that the project Dissertation titled “Graph GUI” which is submitted by us to the Department of INFORMATION TECHNOLOGY, Delhi Technological University, Delhi in partial fulfilment of the requirement for the award of the degree of Bachelor of Technology, is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any Degree, Diploma Associateship, Fellowship or other similar title or recognition.

Place: Delhi Khushi

Date: Himanshi Gupta

**DEPARTMENT OF INFORMATION TECHNOLOGY**

**DELHI TECHNOLOGICAL UNIVERSITY**

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

**CERTIFICATE**

I hereby certify that the Project submitted by Khushi 2K19/IT/067 and Himanshi Gupta 2K19/IT/060; INFORMATION TECHNOLOGY, Delhi Technological University, Delhi in partial fulfilment of the requirement for the award of the degree of Bachelor of Technology, is a record of the project work carried out by the students under my supervision. To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Place: Delhi **Swati Sharda**

Date: 01-12-2020 **SUPERVISOR**

**DEPARTMENT OF INFORMATION TECHNOLOGY**

**DELHI TECHNOLOGICAL UNIVERSITY**

(Formerly Delhi College of

Engineering) Bawana

Road, Delhi-110042

**ACKNOWLEDGEMENT**

I would like to convey our heartfelt thanks to our supervisor Mrs. Swati Sharda for her ingenious ideas, tremendous help and cooperation.

I am extremely grateful to my friends who gave valuable suggestions and guidance for completion of my project. The cooperation and healthy criticism came handy and useful with them.

Finally I would like to thank all the above mentioned people once again.

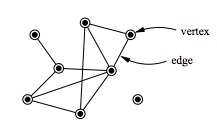
**INTRODUCTION**

A Graph consists of a finite set of vertices (or nodes) and set of Edges which connect a pair of nodes.

A Graph is a non-linear data structure consisting of nodes and edges. The nodes are sometimes also referred to as vertices and the edges are lines or arcs that connect any two nodes in the graph.

A graph ‘G’ is a set of vertex, called nodes ‘v’ which are connected by edges, called links ‘e'. Thus G = (v, e).

**Vertex (Node):** A node v is an intersection point of a graph. It denotes a location such as a city, a road intersection, or a transport terminal (stations, harbours, and airports).  
  
**Edge (Link):** An edge e is a link between two nodes. A link denotes movements between nodes. It has a direction that is generally represented as an arrow. If an arrow is not used, it means the link is bi-directional.



Types of Graphs:

Graphs come in various shapes and forms, however some of the most common types include:

- Undirected graphs

- Directed graphs

- Weighted graphs

- Unweighted graphs

**Undirected graphs:**

Undirected graphs have edges that do not have a direction. The edges indicate a two-way relationship, in that each edge can be traversed in both directions.

**Directed graphs:**

Directed graphs have edges with direction. The edges indicate a one-way relationship, in that each edge can only be traversed in a single direction.

**Unweighted graphs:**

Unweighted graphs have edges that do not have weights, i.e are set to 1. The edges may be bidirectional or unidirectional.

**Weighted graphs:**

Weighted graphs have edges that have weights, the weights can take any

arbitrary values. The edges may be bidirectional or unidirectional.

**APPLICATIONS OF GRAPHS**

Graphs can be used to model many types of relations and processes in physical, biological, social and information systems. Many practical problems can be represented by graphs. Emphasizing their application to real-world systems, the term network is sometimes defined to mean a graph in which attributes (e.g. names) are associated with the vertices and edges, and the subject that expresses and understands the real-world systems as a network is called [network science](https://en.wikipedia.org/wiki/Network_science).

**PROJECT DESCRIPTION**

This project is designed for creating a graphical representation of a weighted graph with nodes and edges. Graph algorithms can also be run on the graph and the results of the algorithms are displayed graphically.

Three algorithms are used in this project i.e. Dijkstra’s Algorithm, Prim’s Algorithm and Bellman Ford’s Algorithm to obtain the Shortest Path and Distance.

The GUI of this project contains the following features:

1. Adding and Removing Nodes

2. Adding and Removing Edges

3. Switching between Algorithms

4. Choice between Graph-types

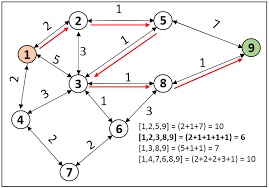
5. Connectedness of a Graph

MODULE: PyQt5

LANGUAGE (Backend and Frontend): Python

**DIJKSTRA’S ALGORITHM**

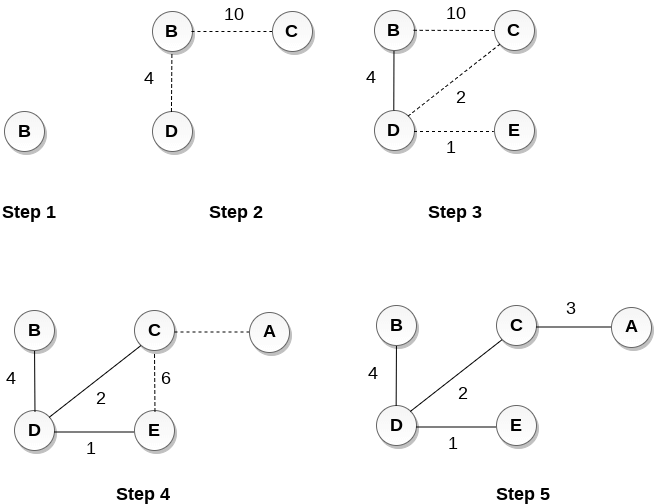
Dijkstra’s algorithm is very similar to [Prim’s algorithm for minimum spanning tree](https://www.geeksforgeeks.org/prims-minimum-spanning-tree-mst-greedy-algo-5/). Like Prim’s MST, we generate a SPT (shortest path tree) with given source as root. We maintain two sets, one set contains vertices included in shortest path tree, other set includes vertices not yet included in shortest path tree. At every step of the algorithm, we find a vertex which is in the other set (set of not yet included) and has a minimum distance from the source.



**PRIM’S ALGORITHM**

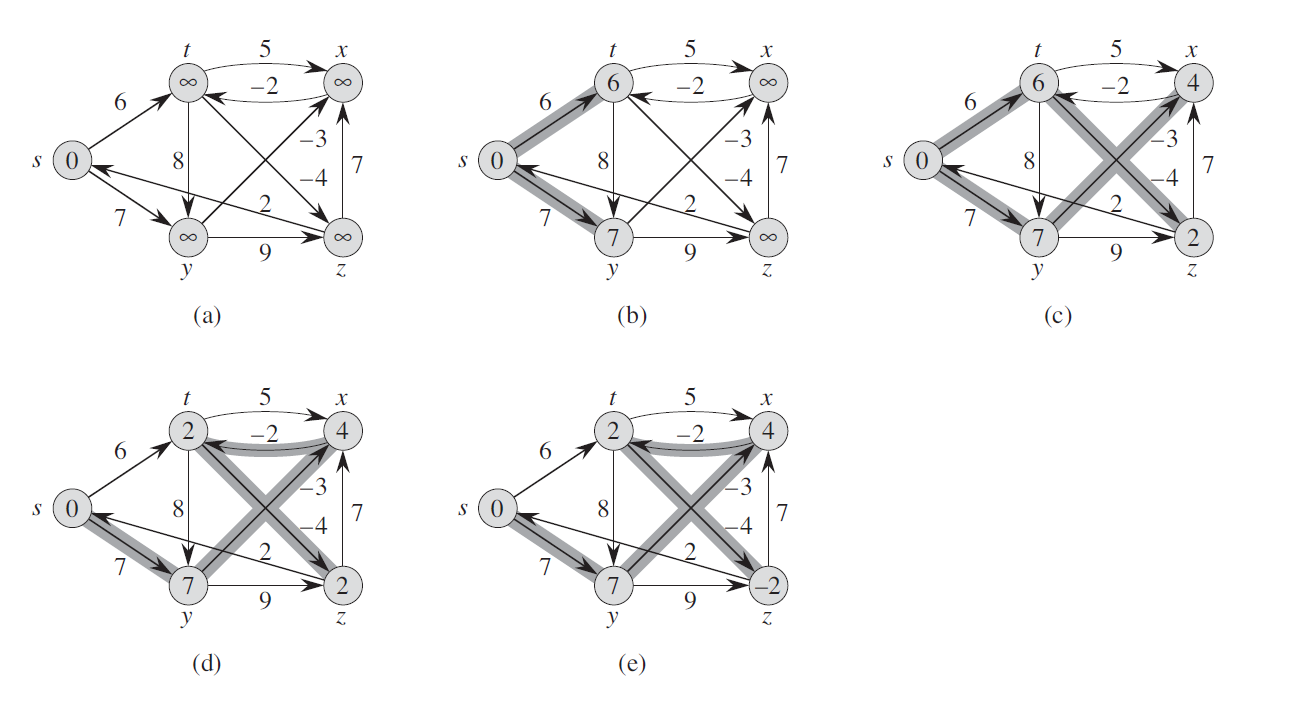
Prim's algorithm to find minimum cost spanning tree (as Kruskal's algorithm) uses the greedy approach. Prim's algorithm shares a similarity with the **shortest path first** algorithms.

Prim's algorithm, in contrast with Kruskal's algorithm, treats the nodes as a single tree and keeps on adding new nodes to the spanning tree from the given graph.



**BELLMAN FORD ALGORITHM**

Dijkstra’s algorithm is a Greedy algorithm and time complexity is O(VLogV) (with the use of Fibonacci heap). Dijkstra doesn’t work for Graphs with negative weight edges, Bellman-Ford works for such graphs. Bellman-Ford is also simpler than Dijkstra and suites well for distributed systems. But time complexity of Bellman-Ford is O(VE), which is more than Dijkstra.



**PyQt MODULE**

PyQt is a GUI widgets toolkit. It is a Python interface for Qt, one of the most powerful, and popular cross-platform GUI library. PyQt is a blend of Python programming language and the Qt library. This introductory tutorial will assist you in creating graphical applications with the help of PyQt.

* A substantial set of [GUI widgets](https://en.wikipedia.org/wiki/GUI_widget)
* [Classes](https://en.wikipedia.org/wiki/Class_(computer_science)) foraccessing [SQL](https://en.wikipedia.org/wiki/SQL) [databases](https://en.wikipedia.org/wiki/Database) ([ODBC](https://en.wikipedia.org/wiki/ODBC), [MySQL](https://en.wikipedia.org/wiki/MySQL), [PostgreSQL](https://en.wikipedia.org/wiki/PostgreSQL), [Oracle](https://en.wikipedia.org/wiki/Oracle_database), [SQLite](https://en.wikipedia.org/wiki/SQLite))
* QScintilla, [Scintilla](https://en.wikipedia.org/wiki/Scintilla_(editing_component))-based rich text editor widget
* Data aware widgets that are automatically populated from a database
* An [XML](https://en.wikipedia.org/wiki/XML) parser
* [SVG](https://en.wikipedia.org/wiki/Scalable_Vector_Graphics) support
* Classes for embedding [ActiveX](https://en.wikipedia.org/wiki/ActiveX) controls on Windows (only in commercial version)

**MAIN FEATURES OF PYQT**

1. Basic 2D plotting in interactive view boxes
2. Image display with interactive lookup tables and level control
3. 3D graphics system
4. Easy to generate new graphics

**TOOLS**

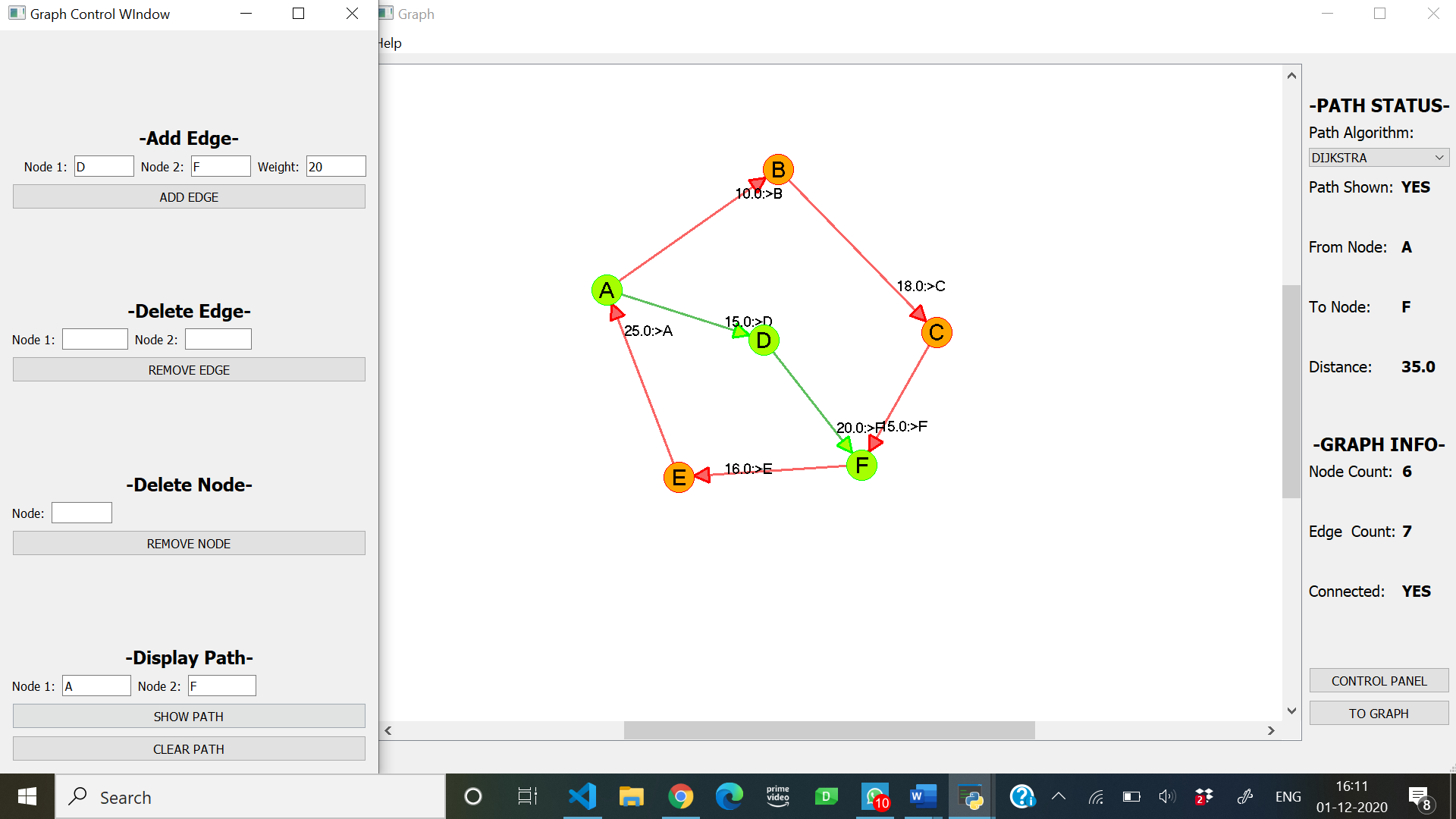
* **QtGui** - Deals with the graphical elements.
* **QtCore** - other non-GUI essentials.
* **QtNetwork** - Networking, as you may have guessed.
* **QtOpenGL** - Allows for the incorporation of OpenGL!
* **QtSql** - Wrapper for SQL handling.
* **QtSvg** - support for support vector graphics.
* **QtXML** - for handling XML data

**CONCLUSIONS**

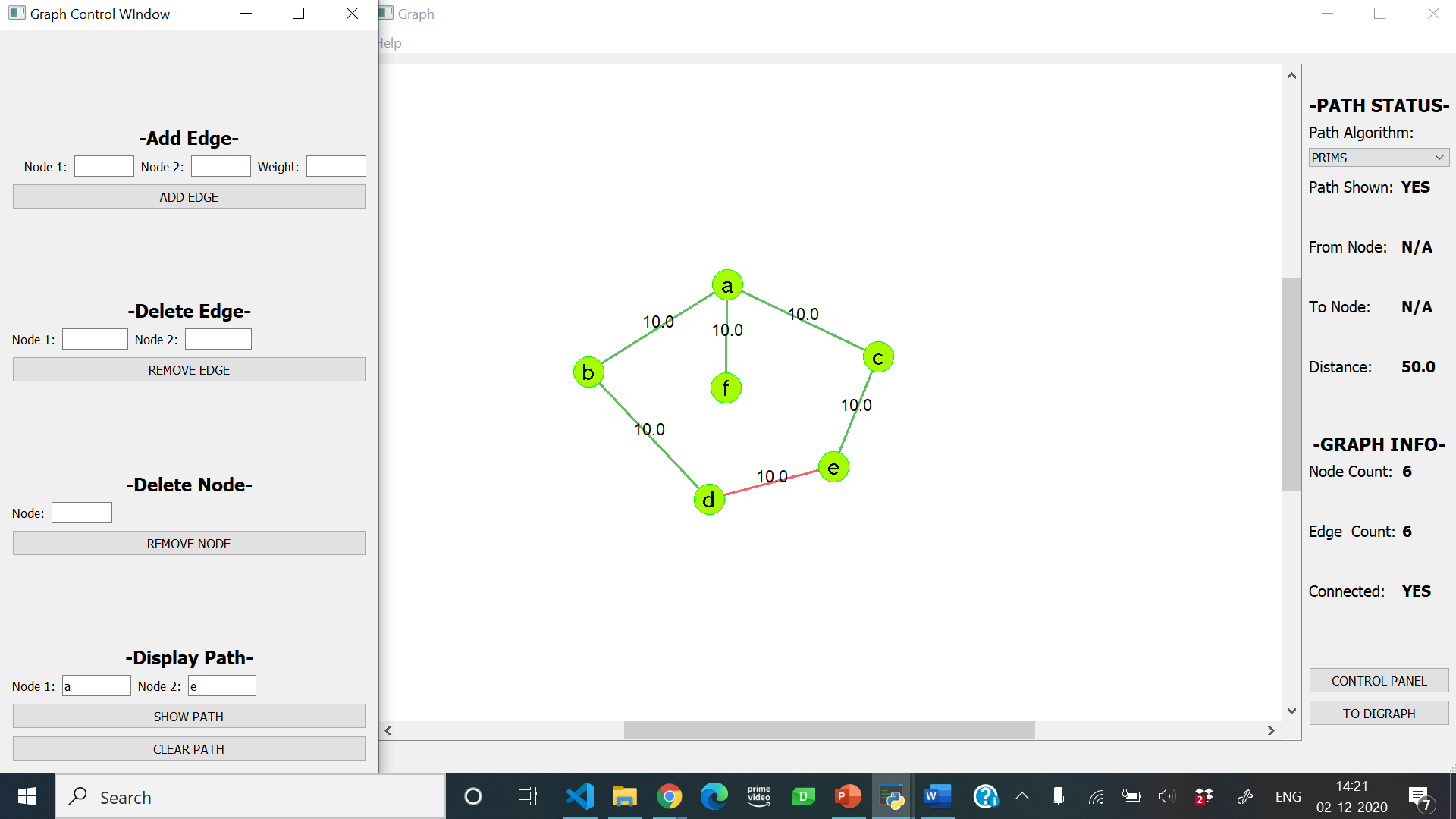
* We were successfully able to make a Graph GUI Application through which the user can learn a lot about Graphs and perform various Algorithms on those graphs.
* In this project we studied and learnt about graphs and various graph algorithms and pyQt5.

**ANNEXURE**

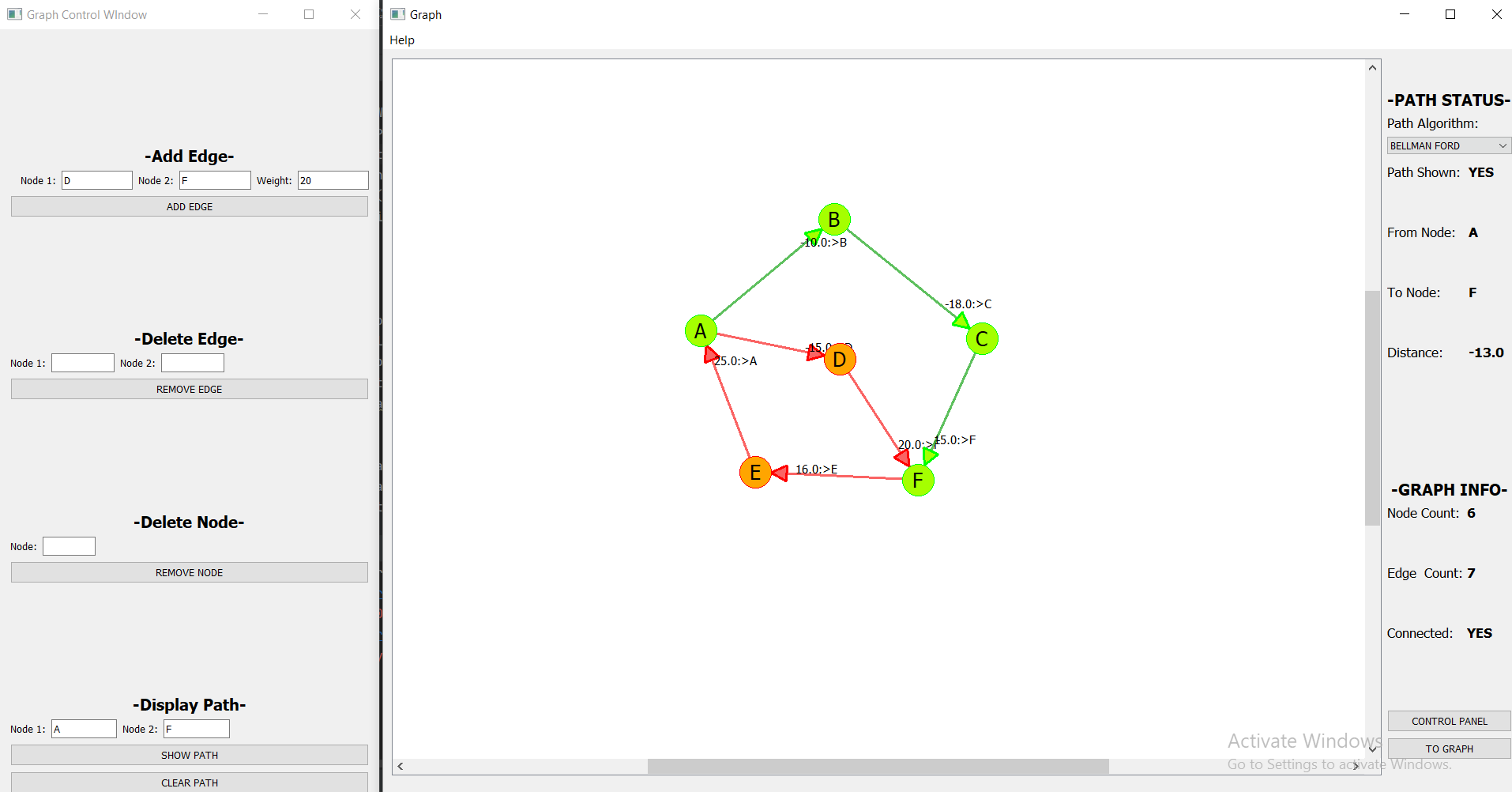
**BY DIJKSTRA’S ALGORITHM**

****

**BY PRIM’S ALGORITHM**



**BY BELLMAN FORD’s ALGORITHM**

****