Title: Graphical User Interface for Graph Manipulation and Analysis

1. Introduction

Overview of the application's purpose: to provide a user-friendly interface for creating, manipulating, and analyzing graphs.

Importance of graph analysis in various fields, including computer science, mathematics, and network design.

2. Application Features

Graph Creation: Allows users to create nodes and edges to form a graph.

Graph Editing: Provides options to add and remove nodes and edges from the graph.

Graph Analysis: Offers various analysis tools to determine properties of the graph, such as Eulerian and Hamiltonian characteristics.

Graph Export/Import: Enables users to save and load graph configurations for later use.

3. Graph Representation

Use of adjacency matrix to represent the graph's structure.

Implementation of graph manipulation methods based on the adjacency matrix representation.

4. User Interface Design

Main Interface Screen: Displays the main interface with options to create, edit, and analyze graphs.

Menu Bar: Provides a menu bar with dropdown options for different graph operations and analyses.

Canvas Display: Utilizes a canvas to visually represent the graph and its elements.

Interactive Features: Supports interactive mouse-driven actions for creating and editing graph elements.

5. Graph Analysis Algorithms

Eulerian Analysis: Implements algorithms to determine if the graph is Eulerian or semi-Eulerian. Offers both Hierholzer's and Fleury's algorithms for finding Eulerian paths or circuits.

Hamiltonian Analysis: Utilizes algorithms to determine if the graph contains a Hamiltonian path or circuit.

6. Usage and Functionality

Demonstration of how users can interact with the application to create, edit, and analyze graphs.

Illustration of various analysis results obtained through the application's features.

7. Conclusion

Recap of the application's capabilities and usability in graph manipulation and analysis tasks.

Potential future enhancements or features that could be added to further improve the application's functionality.

8. References

Mention of any external libraries or algorithms used in the application's development.

Citations for relevant literature on graph theory and analysis algorithms.

This report provides an overview of the graphical user interface developed for graph manipulation and analysis, highlighting its features, functionality, and underlying algorithms.

Title: Serializable Matrix Graph Representation

1. Introduction

Overview of the clsMatrixGraph class: a serializable implementation of a graph using an adjacency matrix.

Explanation of the importance of graph representations in computer science and related fields.

2. Features and Functionality

Adjacency Matrix Representation: Utilizes a 2D array to represent the connections between vertices.

Serialization: Implements Serializable interface for object serialization, allowing the graph to be stored and retrieved from files.

Graph Manipulation: Provides methods for adding and removing nodes and edges, as well as clearing the graph.

Graph Analysis: Supports various graph analysis algorithms, including finding Eulerian and Hamiltonian paths.

3. Eulerian Path Finding

Implementation of Fleury's algorithm to find Eulerian paths or circuits in the graph.

Checks for the existence of an Eulerian path or circuit and returns the path if found.

4. Hamiltonian Path Checking

Utilizes depth-first search (DFS) to check for the existence of Hamiltonian paths in the graph.

Implements a recursive DFS algorithm to traverse the graph and determine if a Hamiltonian path exists.

5. Serialization and Deserialization

Provides methods for exporting the graph to a file using object serialization.

Includes a method for importing a serialized graph from a file.

6. Graph Analysis Methods

Determines if the graph is Eulerian or semi-Eulerian based on connectivity and vertex degrees.

Checks for graph connectivity using DFS traversal.

7. Node and Edge Management

Supports node and edge operations such as adding, removing, and checking for existence.

Manages a list of nodes and their connections in the graph.

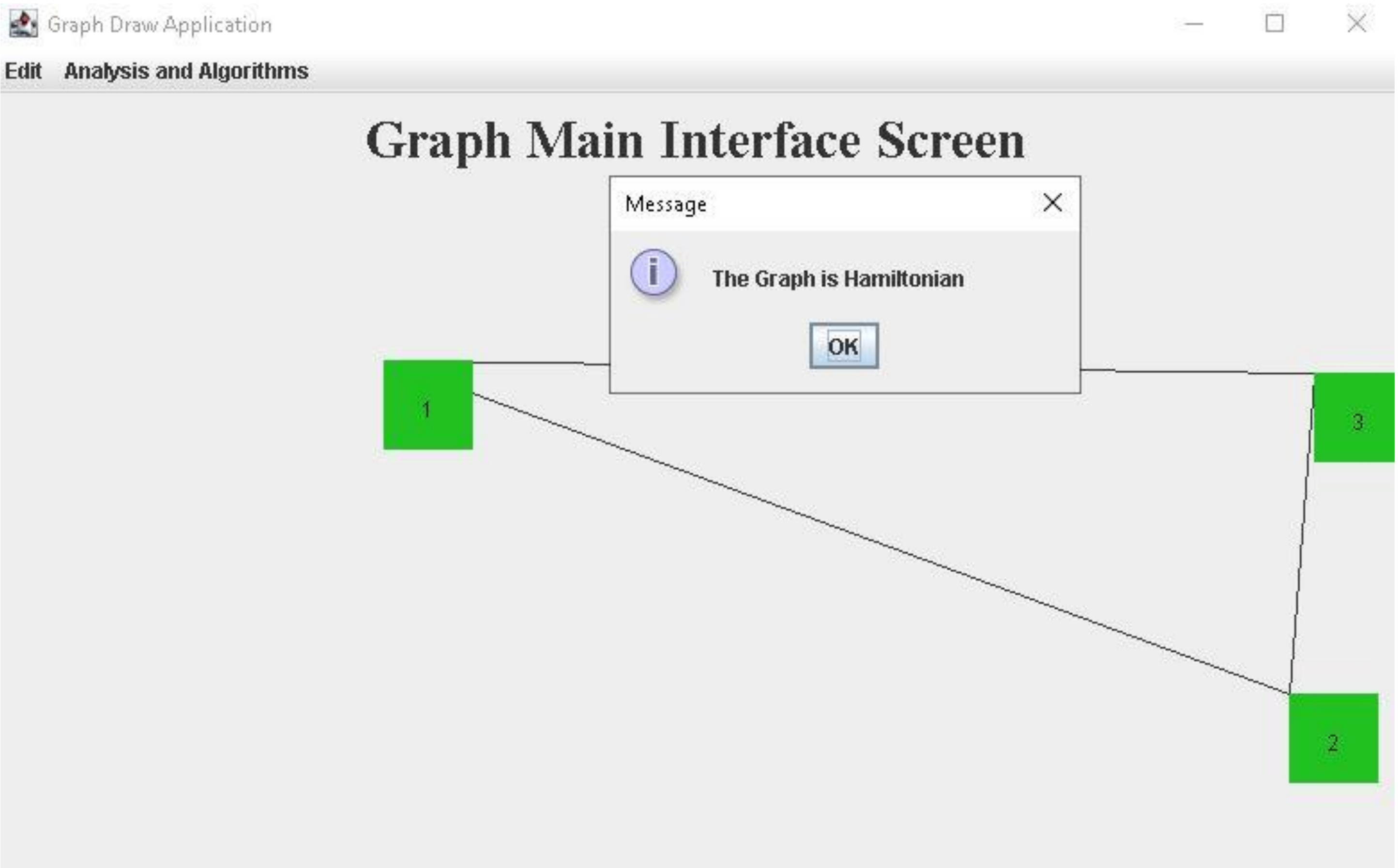
8. Conclusion

Summary of the clsMatrixGraph class's capabilities in representing, manipulating, and analyzing graphs.

Reflection on the significance of graph data structures and algorithms in various applications.

9. References

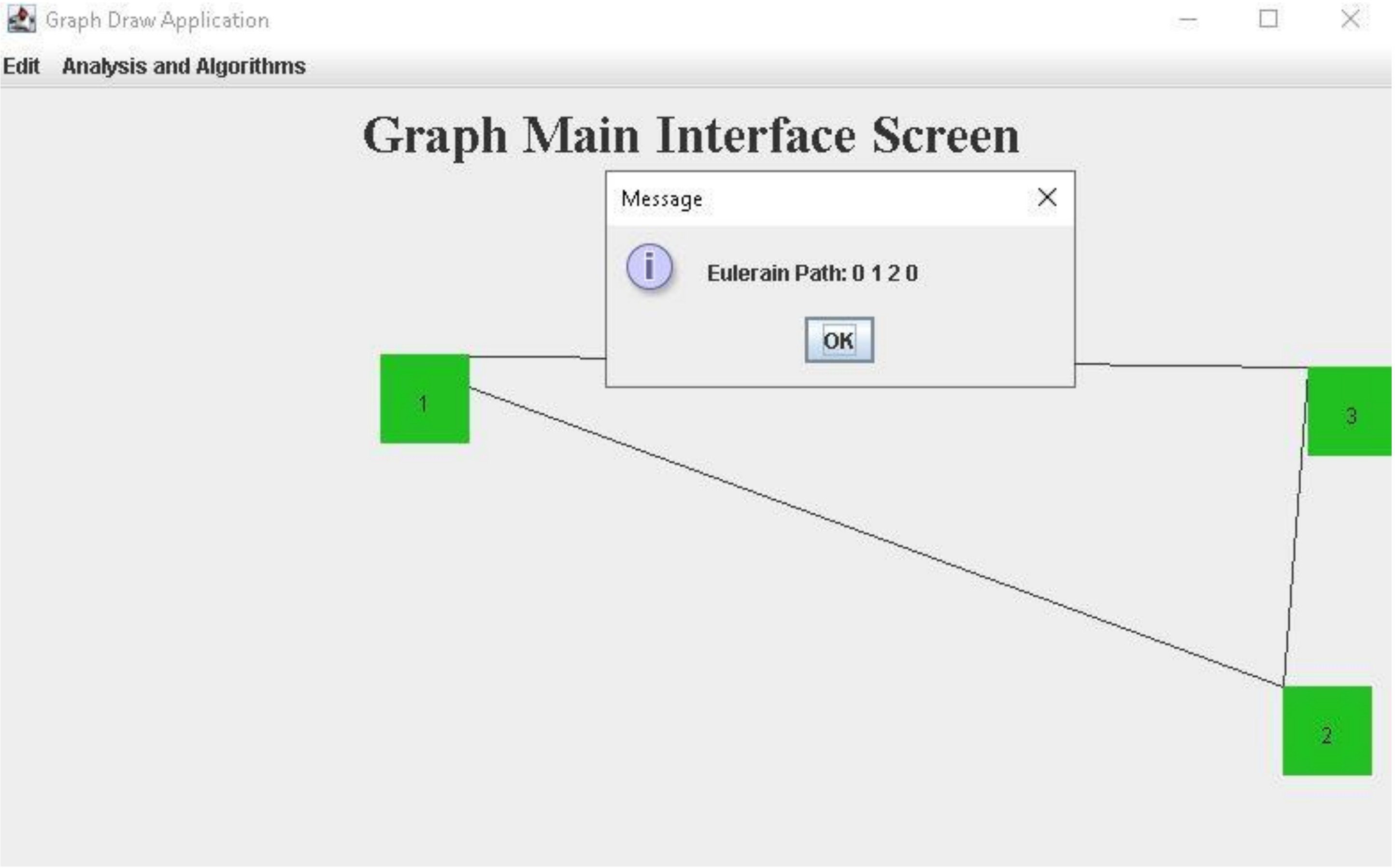
Acknowledgment of any external resources, algorithms, or libraries used in the implementation of clsMatrixGraph.



Graph Main Interface Screen

Message

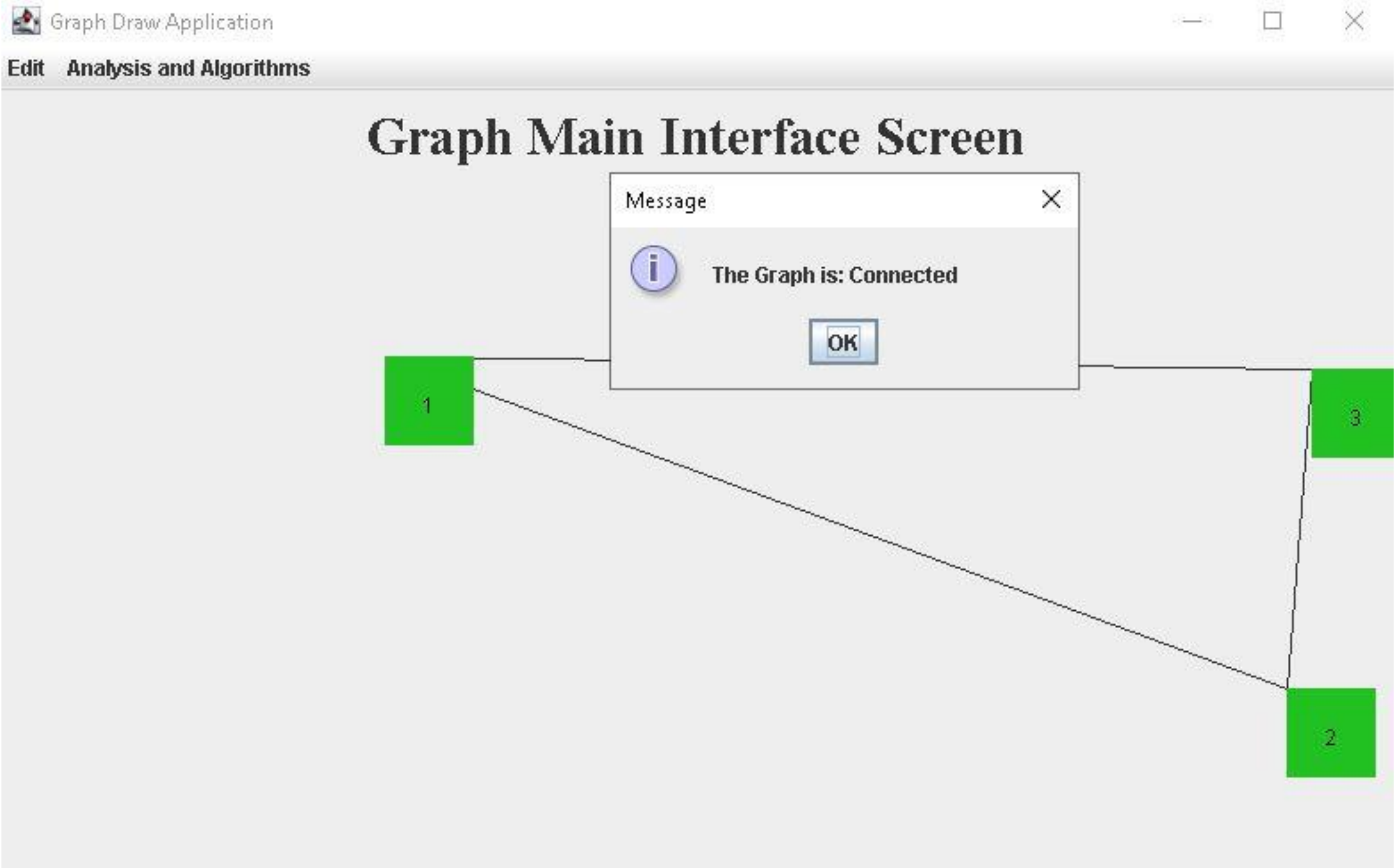
The Graph is Hamiltonian



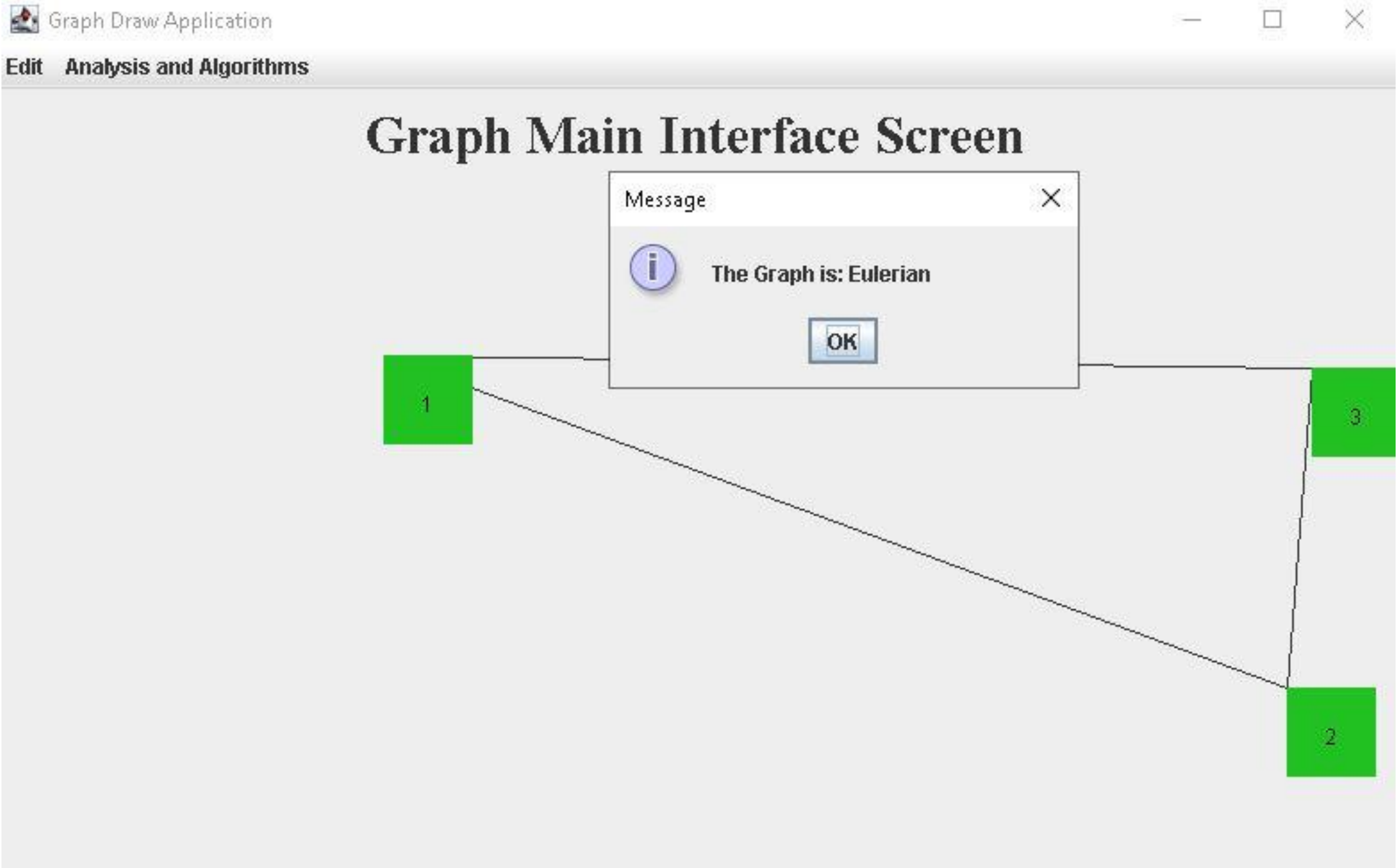
Graph Main Interface Screen

Message

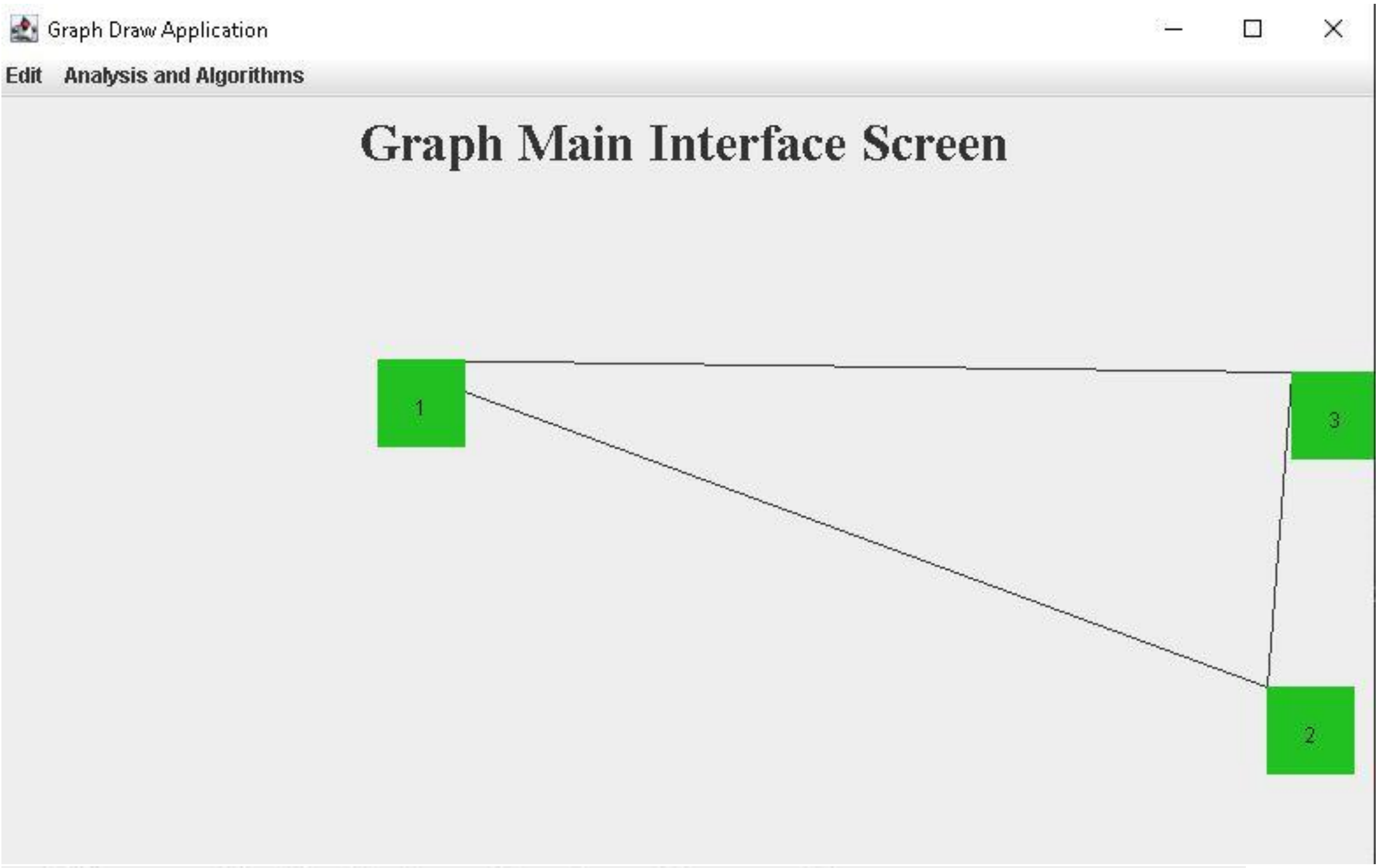
Eulerain Path: 0 1 2 0



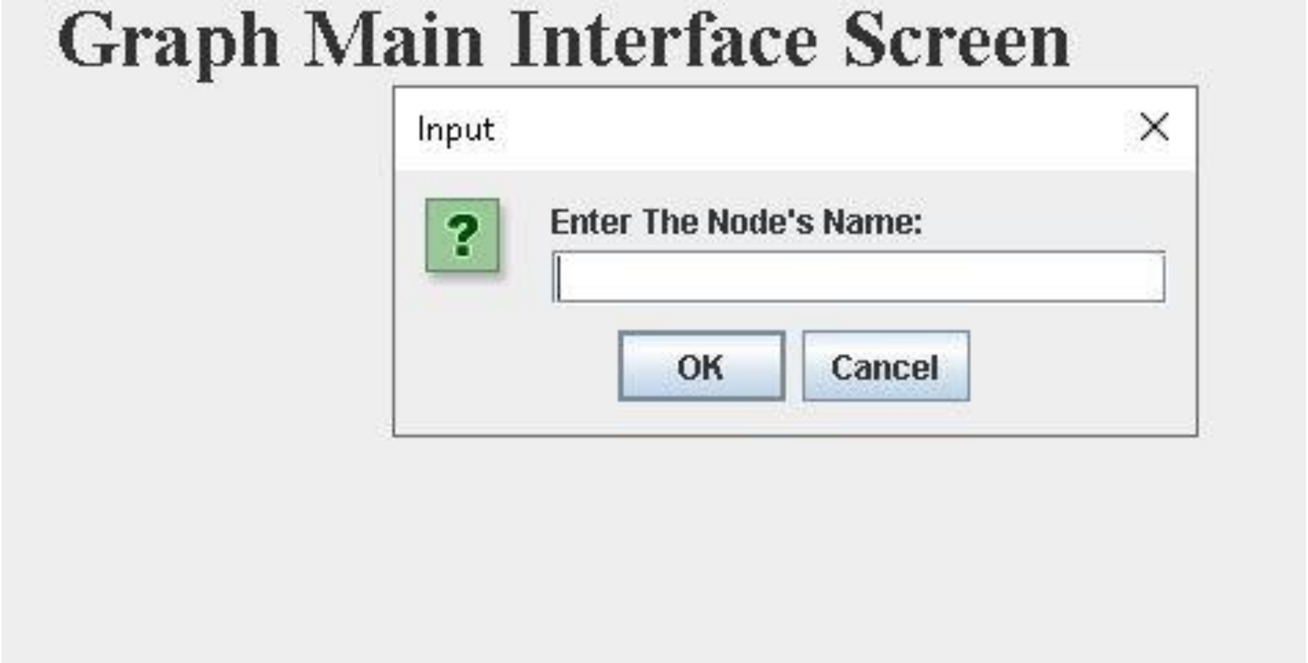
The Graph is: Connected



The Graph is: Eulerian

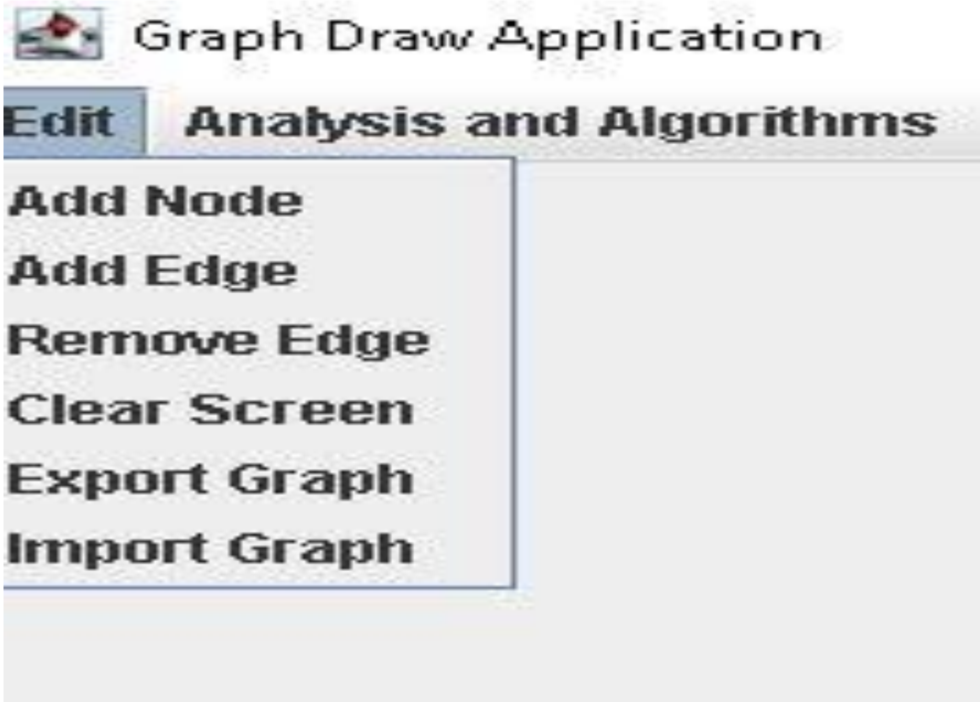


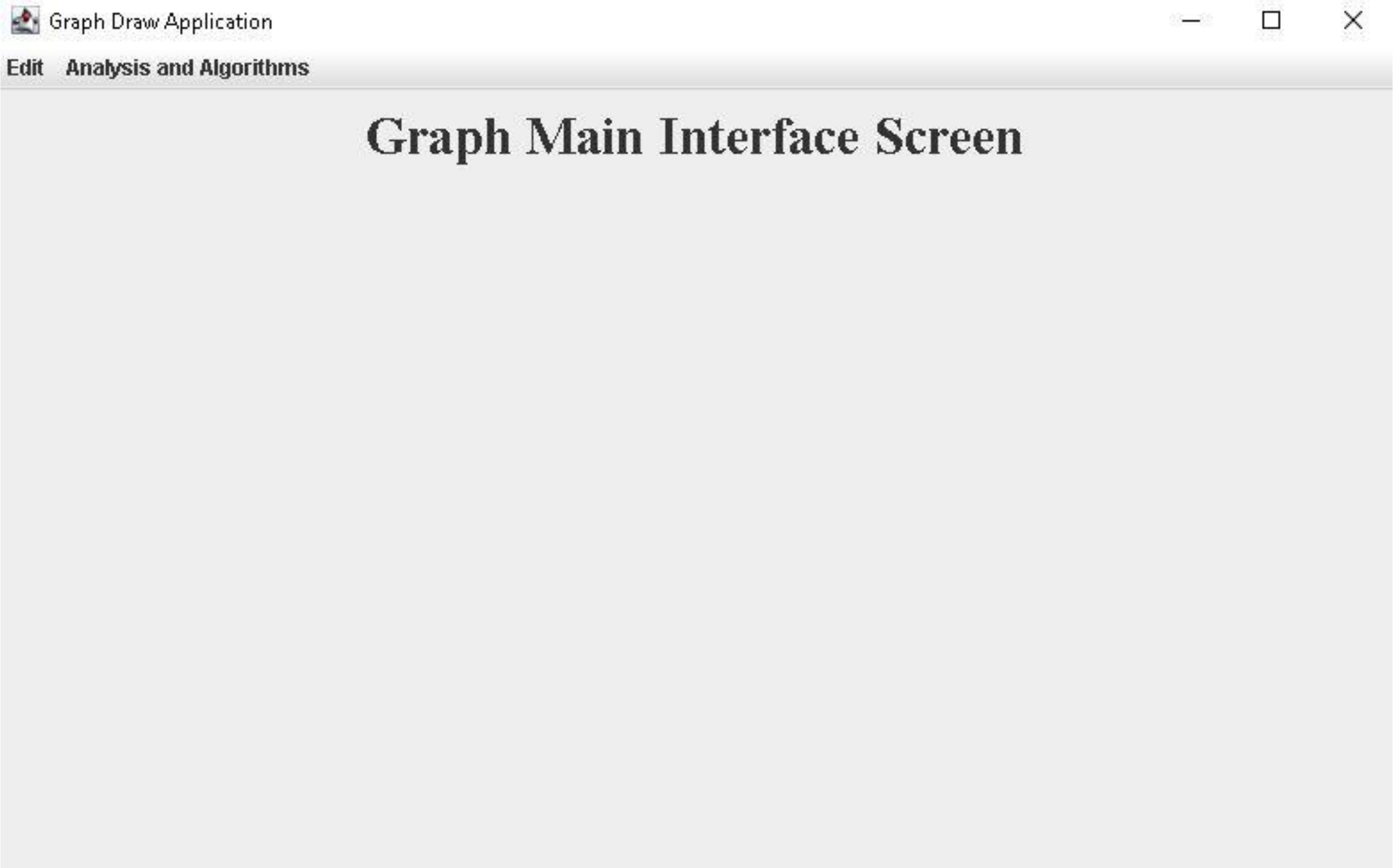
Graph Main Interface Screen



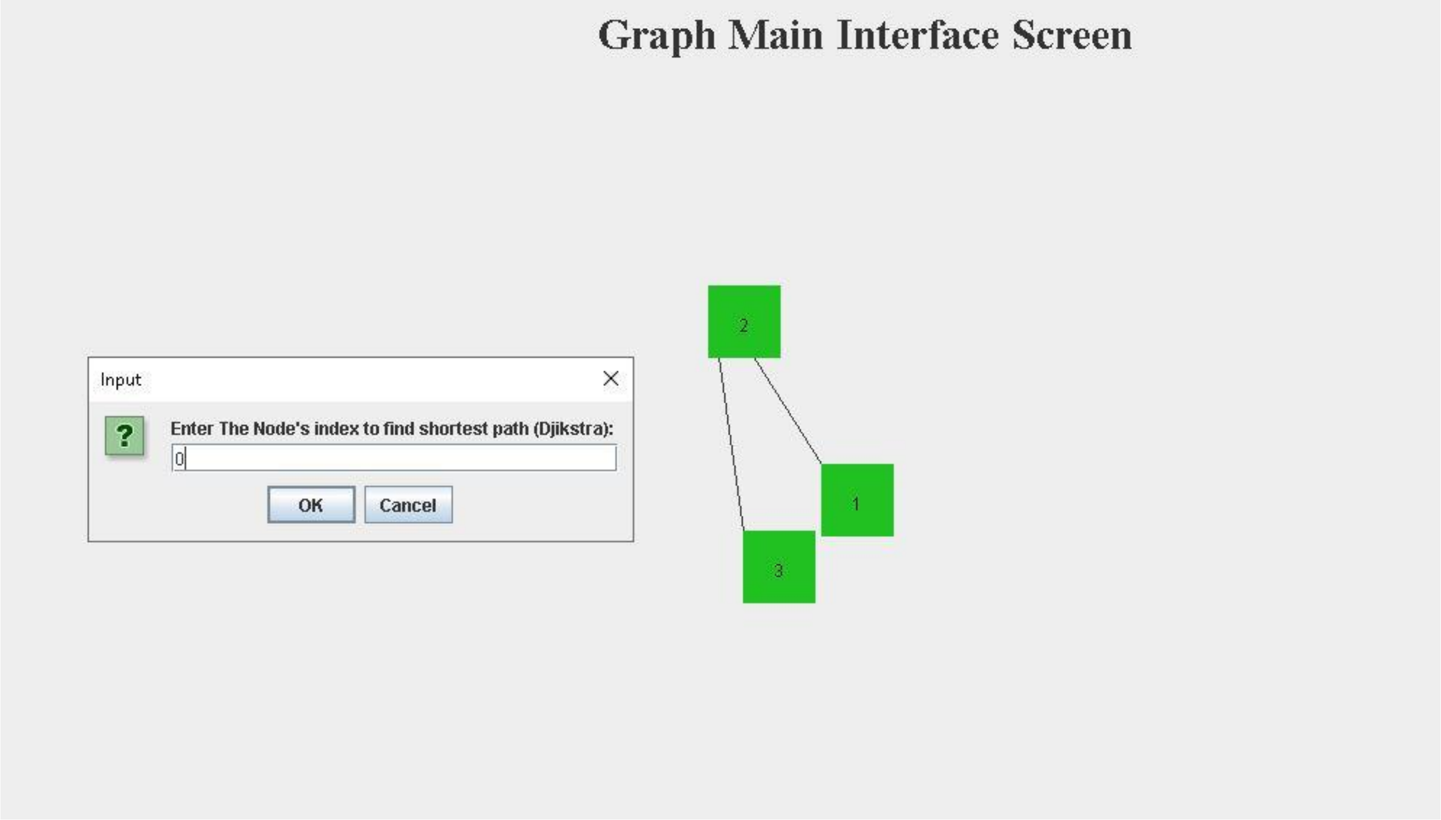
Graph Main Interface Screen

Enter The Node's Name:


Graph Draw Application
Edit
Analysis and Algorithms
Add Node
Add Edge
Remove Edge
Clear Screen
Export Graph
Import Graph

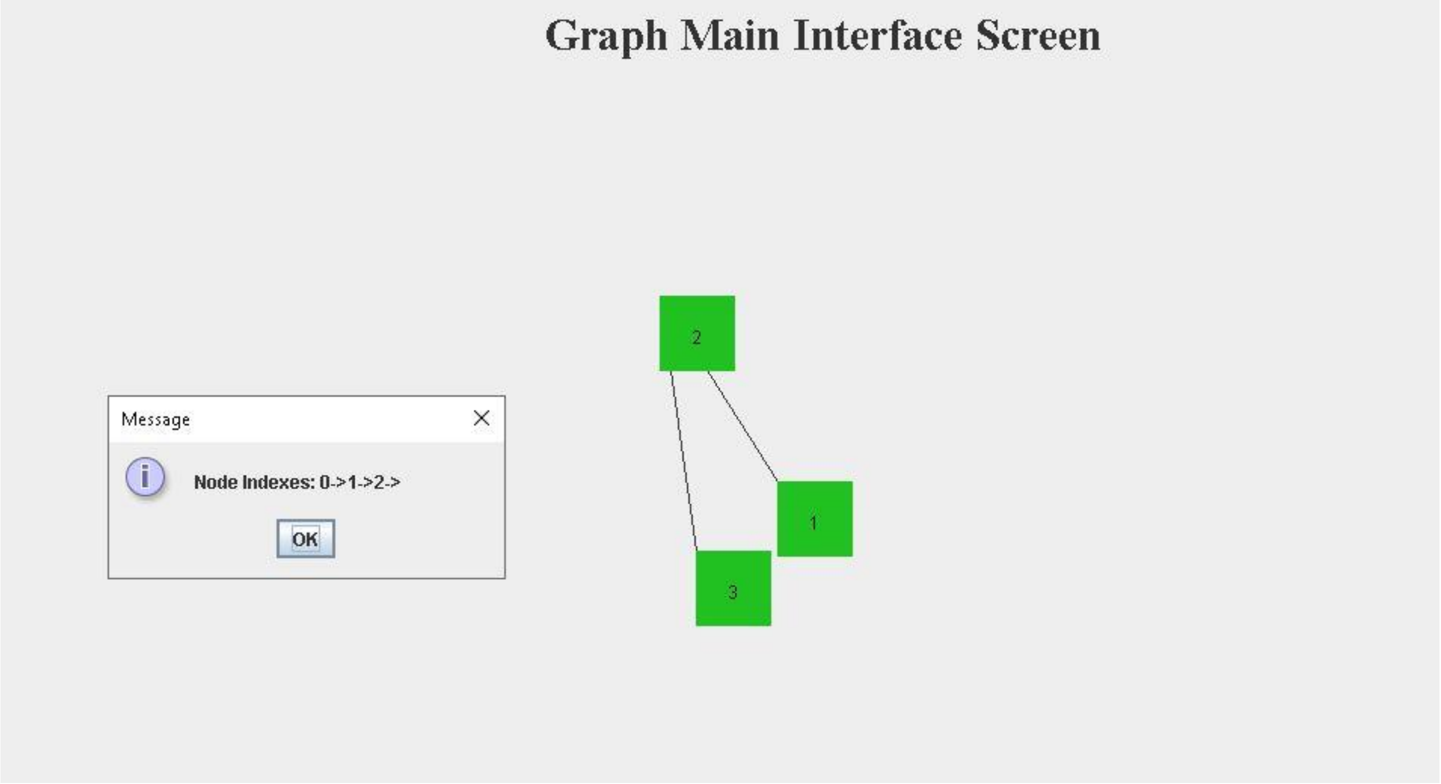

Graph Draw Application
Edit
Analysis and Algorithms
Graph Main Interface Screen

Djikstra



Enter The Node's index to find shortest path (Djikstra):

0



Node Indexes: 0->1->2->