**MATRIX TO GRAPH CONVERTER**

*Dissertation submitted in fulfillment of the requirements for the Degree of*

**BACHELOR OF TECHNOLOGY**

**in**

**COMPUTER SCIENCE AND ENGINEERING**

**By**

**MANAN PANDEY**

**12213157**

Supervisor

**Mr. Shubham Sharma**



**School of Computer Science and Engineering**

Lovely Professional University

Phagwara, Punjab (India)

Month…………… Year ………

@ Copyright LOVELY PROFESSIONAL UNIVERSITY, Punjab (INDIA)

Month ….., Year …..

ALL RIGHTS RESERVED

**ACKNOWLEDGEMENT**

I hereby declare that the project reported in the dissertation/dissertation proposal entitled “**MATRIX TO GRAPH CONVERTER**” in partial fulfilment of the requirement for the award of Degree for Bachelor of Technology in Computer Science and Engineering at Lovely Professional University, Phagwara, Punjab is an authentic work carried out under supervision of my research supervisor Mr. Shubham Sharma. I have not submitted this work elsewhere for any degree or diploma.

I understand that the work presented herewith is in direct compliance with Lovely Professional University’s Policy on plagiarism, intellectual property rights, and highest standards of moral and ethical conduct. Therefore, to the best of my knowledge, the content of this dissertation represents authentic and honest research effort conducted, in its entirety, by me. I am fully responsible for the contents of my dissertation work.

*Signature of Candidate*

**Manan Pandey**

**R.No. K22UN-B-57**

**TABLE OF CONTENTS**

* Introduction of the Project.
* Objectives and Scope of the Project.
* Application tools.
* Methodology / Flowchart or Algorithm Implementation.
* Screenshots of the Execution.
* Summary.
* Bibliography.
* Annexure.

**INTRODUCTION**

In a world driven by data, the ability to represent and analyse complex relationships is paramount. Graph theory, a fundamental branch of computer science, plays a pivotal role in modelling and solving real-world problems where entities and their connections matter. The "Matrix-to-Graph Transformation and BFS Traversal Application" is a versatile Java-based project that bridges the gap between tabular data, specifically matrices, and the realm of graph theory. This application empowers users to convert any type of matrix, be it numeric, textual, or even custom data structures, into a graph representation.

The project's core functionality hinges on two crucial components. First, it employs a custom Graph class with adjacency lists to efficiently store and manage nodes and their connections, embracing the principles of graph theory. Second, it harnesses the power of Breadth-First Search (BFS), one of the most fundamental graph traversal algorithms. Users can initiate BFS from any chosen matrix element, exploring connections through graph nodes in a systematic, level-by-level manner.

Whether it's visualizing social networks, optimizing transportation routes, or uncovering patterns in large datasets, this project offers a tool to tackle a wide array of applications. Its flexibility, robustness, and integration of essential data structures and algorithms make it a valuable asset for researchers, data analysts, and developers seeking to explore relationships in their data from a graph theory perspective. This project serves as a gateway to harness the power of graphs, translating matrices into dynamic networks ripe for analysis and exploration.

**OBJECTIVES AND SCOPE OF THE PROJECT**

**Objectives:**

1. Matrix-to-Graph Conversion : The primary objective of this project is to develop an application that can convert matrices of any data type into graph representations. The application will automate the process of translating data from structured tabular format into dynamic graph structures.

2. Breadth-First Search (BFS) Traversal : The project aims to integrate the BFS algorithm, allowing users to perform efficient graph traversal starting from any specified node. BFS will provide insights into the relationships and connections within the graph.

3. Data Type Agnosticism : This project is designed to be data type agnostic, enabling users to work with matrices containing various types of data, such as integers, strings, custom objects, or any other data structures they might require.

4. Versatility : The application will be versatile and adaptable to a wide range of domains. Its flexibility will make it suitable for diverse applications, including social network analysis, transportation optimization, bioinformatics, recommendation engines, and more.

5. User-Friendly Interface : The project will include a user-friendly interface that simplifies the matrix-to-graph transformation and BFS traversal processes. Users should be able to interact with the application without requiring extensive programming knowledge.

**Scope :**

1. Matrix-to-Graph Transformation :

- The application will accept matrices of varying sizes.

- It will support matrices with elements of any data type.

- Matrices can be manually input or imported from external data sources.

2. Graph Representation :

- The project will use an adjacency list data structure for graph representation.

- It will automatically generate edges between nodes based on the positions of elements in the matrix.

- Users will have the option to visualize the resulting graph.

3. Breadth-First Search (BFS) :

- BFS traversal will be implemented to explore the graph, starting from a user-specified node.

- Users will have the option to view the BFS traversal results, which can include node sequences and any related information.

4. Versatility Across Domains :

- The application will be domain-agnostic and can be applied to various fields, including but not limited to social networks, transportation systems, bioinformatics, recommendation engines, and more.

5. User Interface :

- The project will feature a graphical user interface (GUI) to make it accessible to a wide range of users.

- The GUI will include input forms for matrix data, options for BFS initiation, and visualization of the transformed graph.

6. Customization and Extensibility :

- The project will offer options for customizing BFS traversal algorithms to suit specific use cases.

- It will be designed with extensibility in mind, allowing users to add new functionalities or integrate additional algorithms.

7. Documentation and Support :

- Comprehensive documentation will be provided to assist users in understanding and using the application effectively.

- Users will have access to support resources, such as user manuals and tutorials.

This project's objectives and scope encompass creating a versatile and user-friendly application that empowers individuals across various domains to analyse data more effectively by converting matrices into graph structures and performing BFS traversals. It encourages exploration and insights in a wide range of applications while maintaining adaptability and ease of use.

**APPLICATION TOOLS**

1. Java Programming Language :

- Java serves as the primary programming language for the application's development. Its robust, platform-independent nature, extensive standard library, and object-oriented features are crucial for creating the core logic of the application.

2. Integrated Development Environment (IDE) :

- An Integrated Development Environment is essential for Java application development. It is used for writing, debugging, and managing the Java code. We have used IntelliJ Idea for this project.

3. Graph Representation :

- The project implements graph representation using DSA concepts. An adjacency list is used to store and manage graph nodes and their connections. This is a fundamental DSA structure for graph representation.

4. Breadth-First Search (BFS) :

- The BFS algorithm is a key DSA component used to traverse the graph efficiently. It enables the exploration of relationships and connections within the graph in a systematic manner.

5. Custom Data Structures :

- DSA concepts are employed to create custom data structures that are agnostic to data types. These data structures facilitate the storage and manipulation of various data types, such as integers, strings, or custom objects, within the application.

**METHODOLOGY**

1. User Input :

- The application starts by accepting user input, including the matrix data to be converted into a graph, the type of data, and the starting node for BFS traversal. This input is entered through the graphical user interface.

2. Matrix-to-Graph Conversion :

- The application proceeds to convert the input matrix into a graph representation using the adjacency list data structure. It follows these steps:

- Iterate through the matrix elements.

- For each element, create a corresponding node in the graph.

- Connect the nodes by establishing edges based on their positions in the matrix (i.e., connecting adjacent nodes).

- Build the graph representation.

3. Breadth-First Search (BFS) :

- After the graph is constructed, the application initiates a BFS traversal. The BFS algorithm is used to explore the graph efficiently:

- Start the BFS traversal from the user-specified starting node.

- Use a queue data structure to keep track of nodes to visit.

- Continue the traversal until all reachable nodes are explored.

- Store and display the order in which nodes are visited during BFS.

4. User Interaction and Visualization :

- The application provides user-friendly interaction through the GUI, allowing users to input data, initiate BFS, and view traversal results. Additionally, users may have the option to visualize the graph, depending on the project's complexity and requirements.

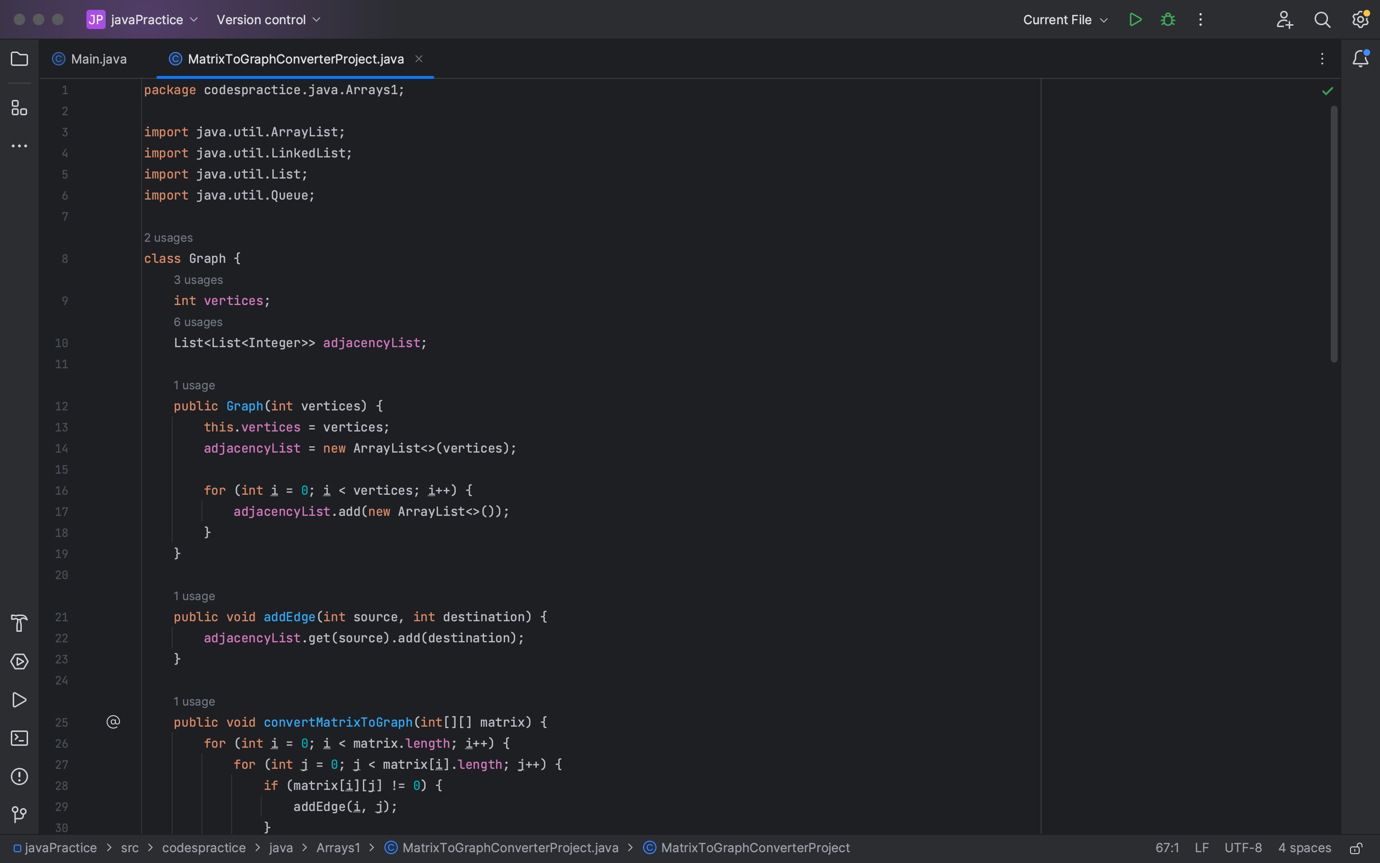
5. Optional Data Analysis (Domain-Specific) :

- Depending on the project's scope and domain, the application can include additional functionalities for data analysis, such as identifying connected components, calculating distances, or performing other relevant operations.

**SCREENSHOTS OF THE CODE AND IT’S EXECUTION**

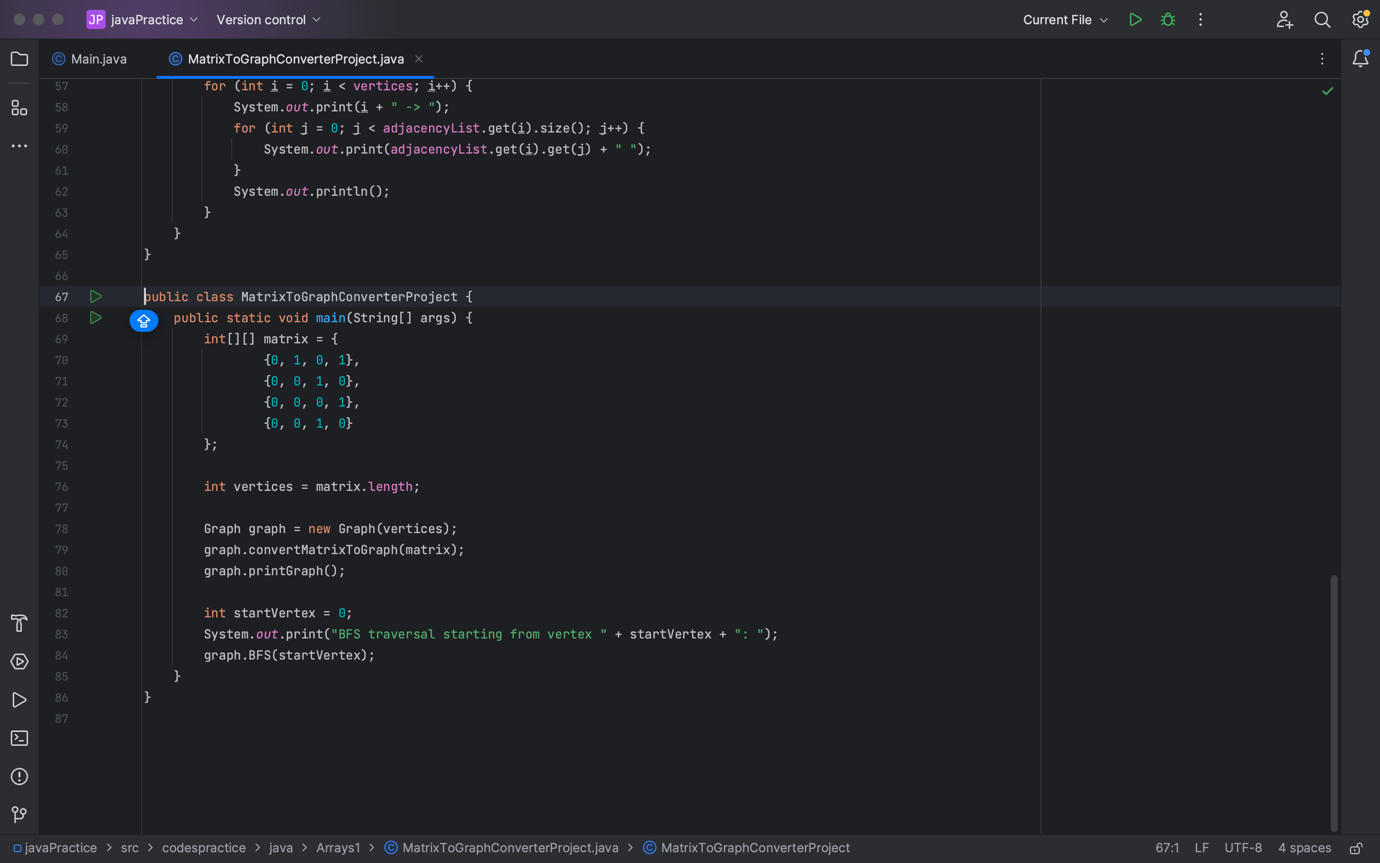
In this project, I have converted a Matrix to a Graph and then performed the BFS (Breadth-First Search) algorithm on the matrix after it has been converted into a node in the graph.

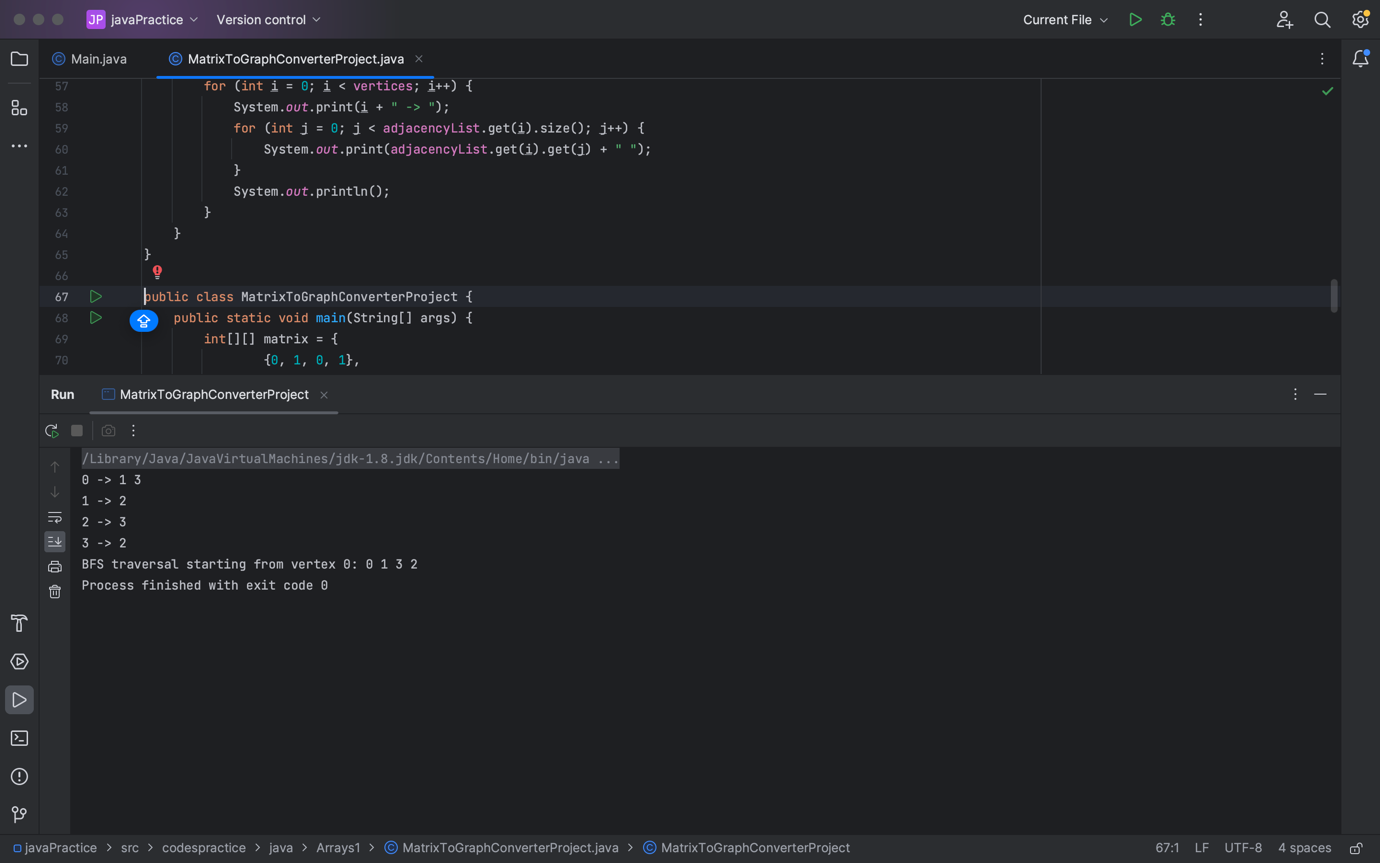
The screenshots of the code and the final output are attached below :

****

**A screen shot of a computer

Description automatically generated**

****

****

**SUMMARY**

The "Matrix-to-Graph Transformation and BFS Traversal Application" is a Java-based tool that bridges the gap between structured data in the form of matrices and the powerful realm of graph theory. The project's primary aim is to enable users to effortlessly convert matrices of various data types into graph representations, a process essential for uncovering complex relationships and patterns within data.

The application excels in its adaptability, accommodating data types that span integers, strings, and custom objects, fostering an inclusive approach to data analysis. It boasts a user-friendly graphical interface, making it accessible to a wide range of users, regardless of their programming expertise. Users can input their matrix data, specify the data type, and initiate a Breadth-First Search (BFS) traversal, all through the intuitive GUI.

Under the hood, the application employs robust data structures and algorithms. It leverages an adjacency list-based graph representation, a fundamental data structure in graph theory, to meticulously create and store graph nodes and their relationships. The BFS algorithm, a staple in graph traversal, is employed to explore these graph structures systematically, starting from a user-defined node and unearthing intricate connections.

Whether in the domains of social network analysis, transportation optimization, bioinformatics, recommendation systems, or countless other fields, this project offers a versatile tool for users to embark on data-driven explorations. It simplifies the process of transforming matrices into graphs and facilitates the systematic traversal of these graphs, empowering users to gain insights, optimize decision-making, and unlock the hidden potential of their data.

In summary, this project harmonizes Java programming, data structures, algorithms, and user-centric design to provide a comprehensive solution for translating structured data into dynamic networks ready for analysis and exploration, and thus, serves as a versatile asset in the data scientist's toolkit.

**BIBLIOGRAPHY**

Books Used :

1. Horstmann, C. S. (2018). "Core Java Volume I--Fundamentals." Pearson.

2. Koffman, E., & Wolfgang, P. (2010). "Data Structures: Abstraction and Design Using Java." Wiley.

3. Weiss, M. A. (2013). "Data Structures and Algorithm Analysis in Java." Pearson.

Online Resources. :

1. GeeksforGeeks. [https://www.geeksforgeeks.org/](https://www.geeksforgeeks.org/)

2. Oracle Java Documentation. [https://docs.oracle.com/en/java/](https://docs.oracle.com/en/java/)

3. NetworkX Documentation. [https://networkx.github.io/](https://networkx.github.io/)