**National University of Computer & Emerging Sciences**

**Karachi Campus**



**TITLE OF PROJECT**

**Project Report**

**Graph Theory**

**Section: BCS-5B**

**Group Members:**

**22K-4493 Azka Sahar Shaikh**

**22K-4364 Muhammad Sudais**

**22K-4201 Sumaiya Waheed**

# **Project : Nearest Neighbor,Nearest Insertion and Brute Force Algorithm for Traveling Salesman Problem (TSP) Using Graph Theory**

The Traveling Salesman Problem (TSP) is a well-known combinatorial optimization problem that focuses on finding the shortest possible route for a salesman to visit a given set of cities exactly once and return to the starting point. In this project, we implemented the TSP using three approaches: Nearest Neighbor, Nearest Insertion, and Brute Force. The project was developed using Python and C++, with the Python implementation featuring a user-friendly GUI created using the customtkinter library for visualization.

#### **Objective**

The primary objective of this project was to understand and compare the performance of various algorithms for solving the TSP in terms of computational efficiency, accuracy, and usability. The GUI was designed to allow users to interact with the problem visually, enhancing comprehension of the algorithms' working principles.

#### **Implementation**

1. Nearest Neighbor Algorithm:  
   This greedy approach starts from a chosen city and iteratively selects the nearest unvisited city. While it is computationally efficient, it may not always yield the optimal solution.
2. Nearest Insertion Algorithm:  
   This heuristic constructs a tour by initially connecting two cities and gradually inserting other cities into the tour at positions that minimize the increase in total distance. It provides a more refined solution compared to Nearest Neighbor.
3. Brute Force Algorithm:  
   This method exhaustively computes all possible permutations of city visits to find the shortest path. Though it guarantees the optimal solution, it is computationally expensive and feasible only for a small number of cities.

#### **Features**

* Python Implementation: The Python code integrates a customtkinter GUI, allowing users to input city coordinates or generate random datasets. The GUI dynamically visualizes the solution paths for the three algorithms, offering an intuitive understanding of the results.
* C++ Implementation: The C++ version emphasizes performance optimization and provides a command-line interface for algorithm execution and comparison.

#### **Results and Observations**

The project highlighted the trade-offs between the algorithms:

* **Nearest Neighbor** performed well for smaller datasets but was less accurate for larger ones.
* **Nearest Insertion** provided a balance between computational speed and solution quality.
* **Brute Force** was accurate but impractical for datasets with more than 10-12 cities due to exponential time complexity.

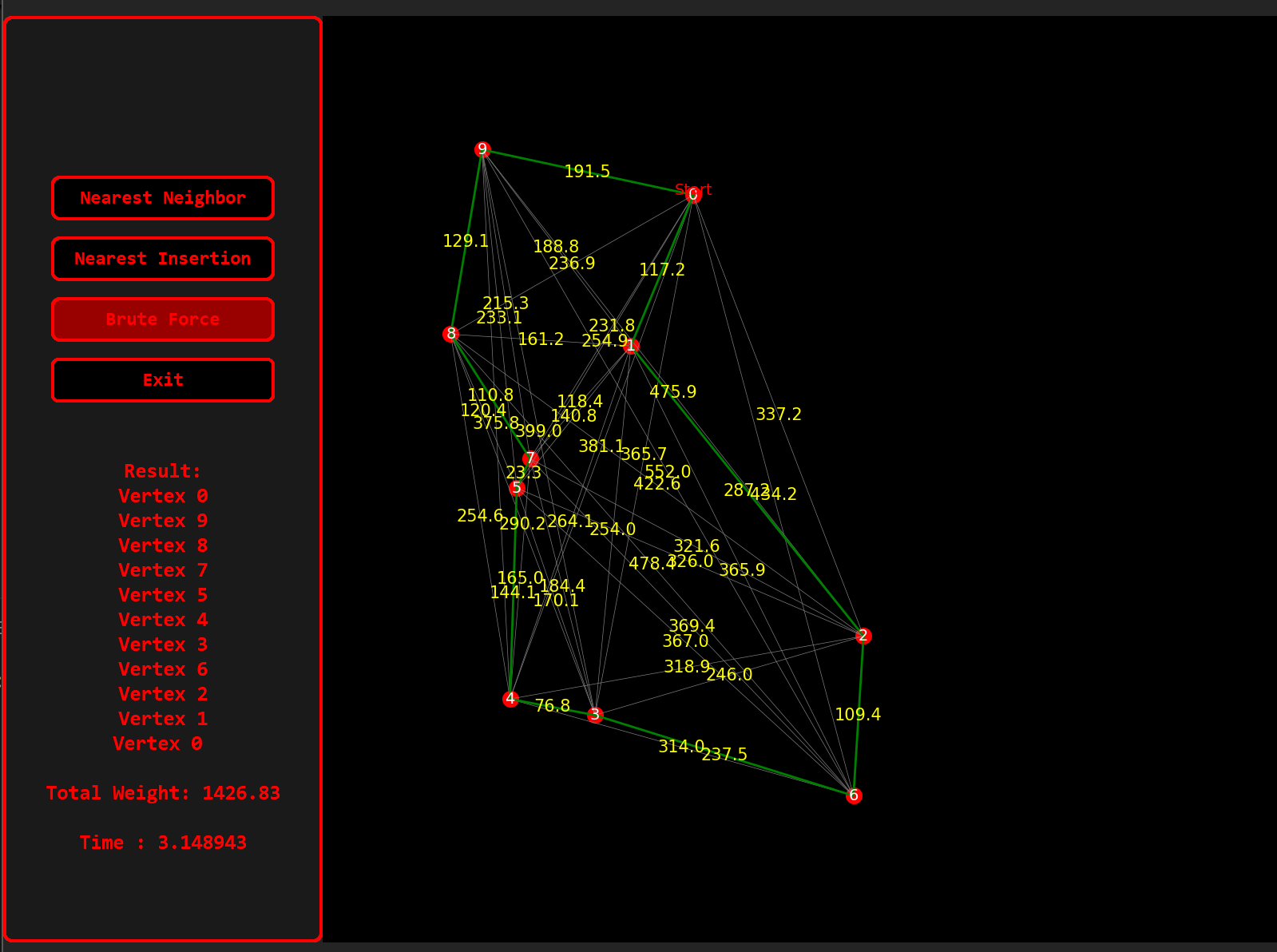
**Language Comparision**

* C++: C++ is faster because it is compiled, meaning it turns your code into machine instructions that the computer understands directly. This makes it very quick when performing heavy calculations, like solving the Traveling Salesman Problem (TSP).
* Python: Python is interpreted, meaning it reads and runs your code line by line. This takes longer because there's extra work involved in managing the code while it runs. Python also uses more memory, which can slow things down, especially with complex algorithms.
* Python is easier to learn with simpler syntax and automatic memory management. It’s slower in execution but great for web apps, data analysis, AI, and rapid development due to its vast libraries. It’s portable and beginner-friendly.
* C++ is faster and provides more control over memory and system resources, making it ideal for performance-heavy tasks like game development and embedded systems. It has a steeper learning curve and requires manual memory management, but it offers greater control and efficiency.

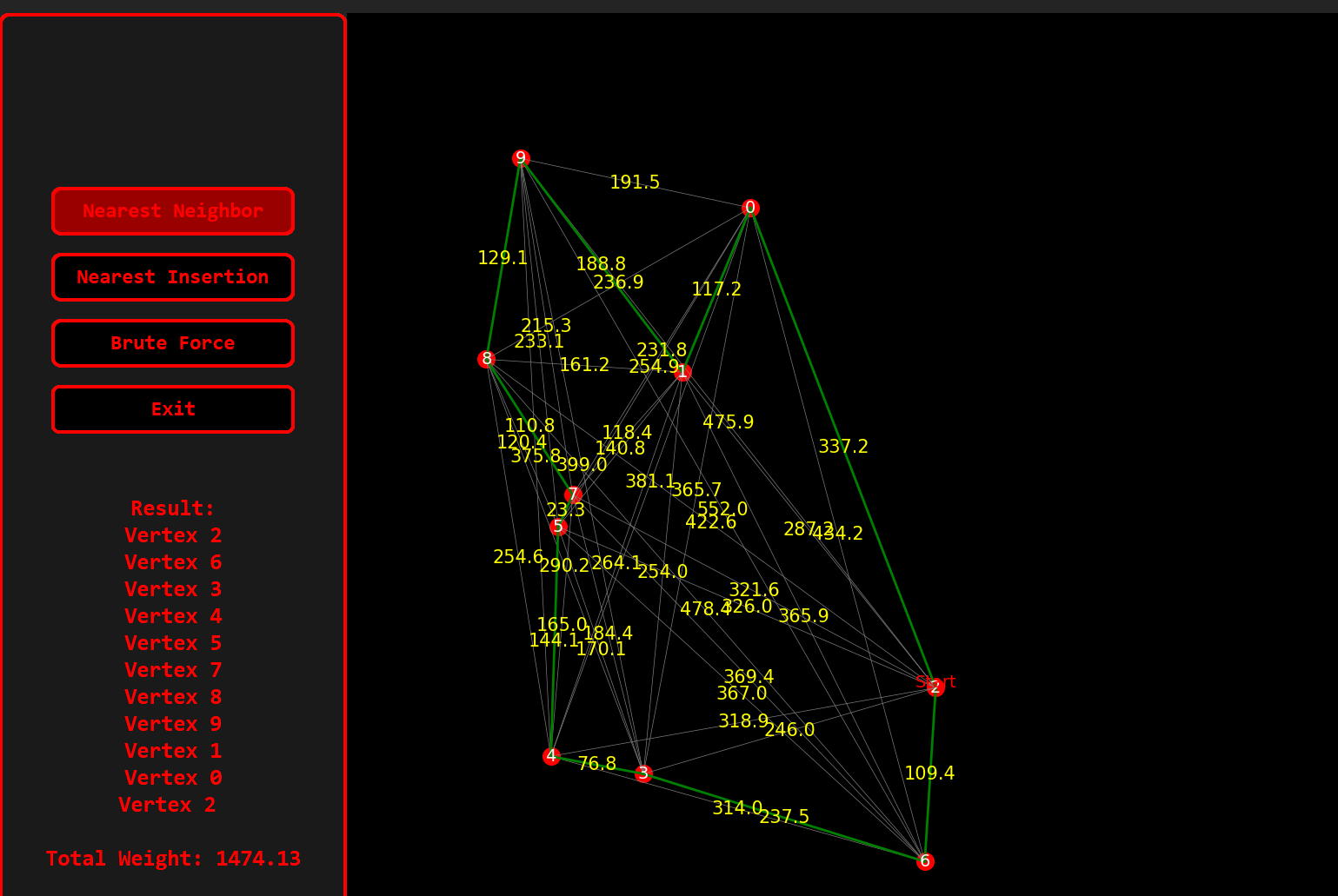
**Project Outputs**

**Python:**

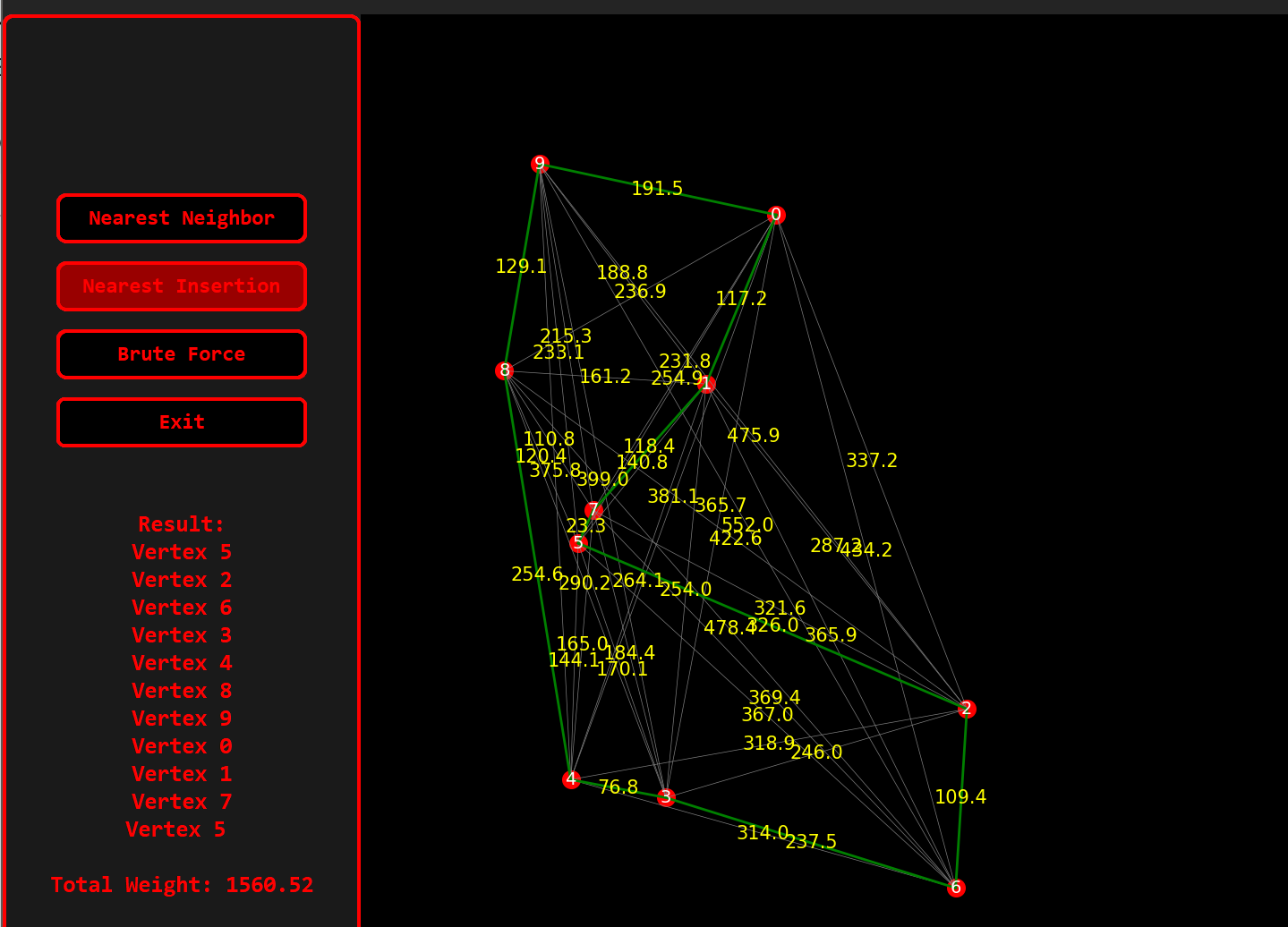
* **Brute force**

****

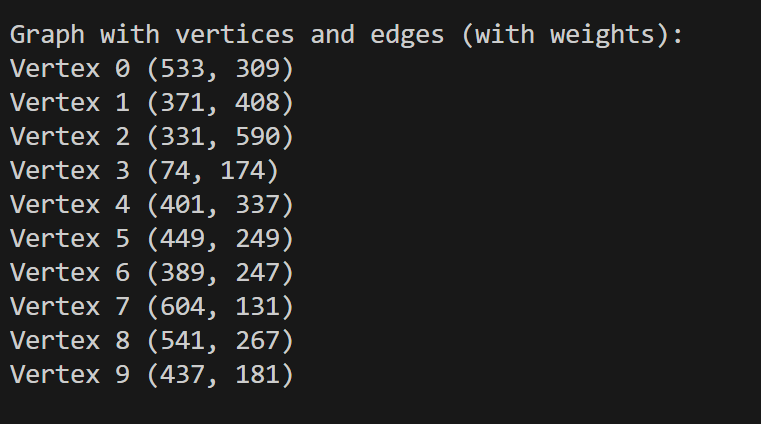
* **Nearest Neighbor**

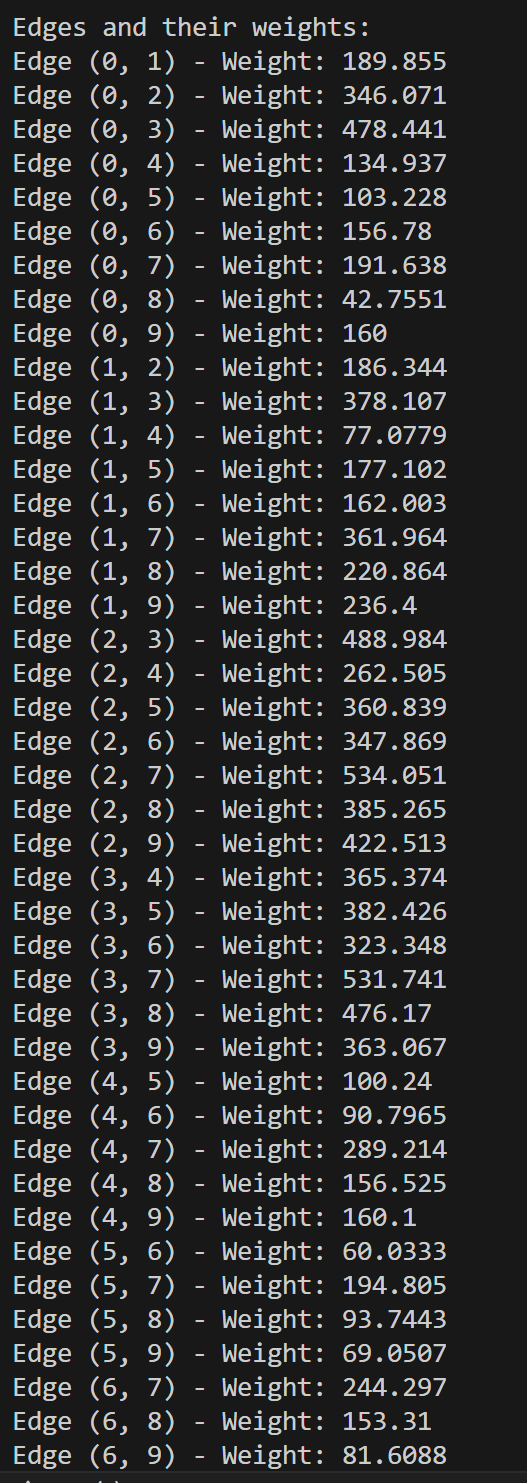
****

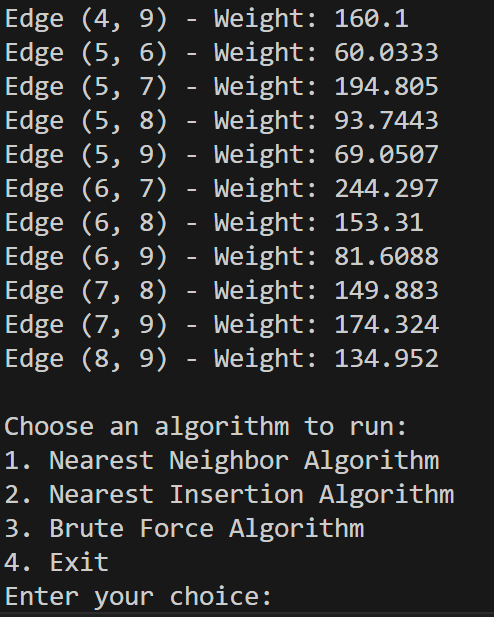
* **Nearest Insertion**

****

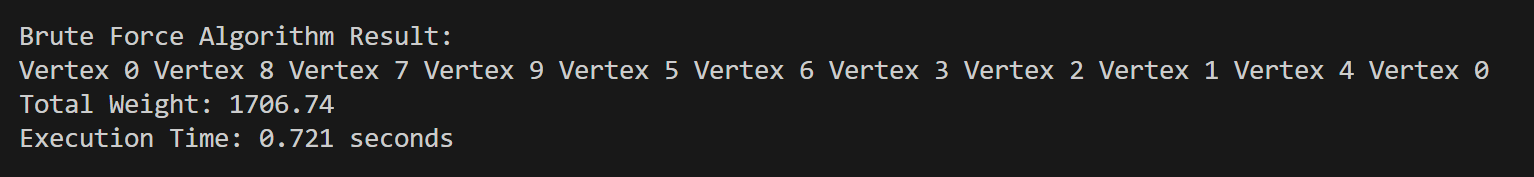
**C++:**

****

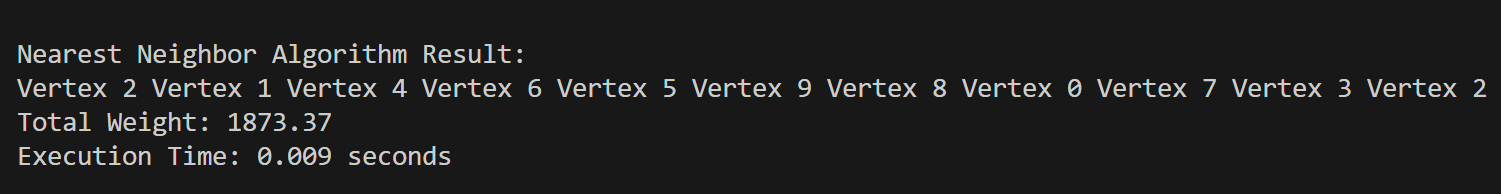
****

****

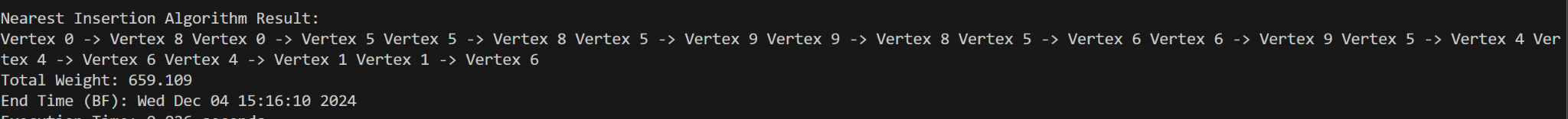
* **Brute Force**

****

* **Nearest Neighbor**

****

* **Nearest Insertion**

****

#### **Conclusion**

This project successfully demonstrated the strengths and limitations of different TSP algorithms through a practical, comparative study. The GUI enhanced user interaction, making the learning process engaging and informative. Future work could include integrating more advanced algorithms like Genetic Algorithms or Simulated Annealing and extending the GUI's capabilities for larger datasets and real-world applications.