**National University of Computer & Emerging Sciences**

**Karachi Campus**



**TITLE OF PROJECT**

**Project Proposal**

**Graph Theory**

**Section: BCS-5B**

**Group Members:**

**22K-4493 Azka Sahar Shaikh**

**22K-4364 Muhammad Sudais**

**22K-4201 Sumaiya Waheed**

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

Group Members:

Member 1:[Muhammad Sudais] - Student ID: [22k-4364]

Member 2: [Azka Sahar Shaikh] - Student ID: [22k-4493]

Member 3: [Sumaiya Waheed] - Student ID: [22k-4201]

## Objective

The primary objective of this project is to evaluate and implement graph traversal algorithms and explore their applications in Computer Science. This project focuses on solving the Traveling Salesman Problem (TSP) using three algorithms:

* **Nearest Neighbor**
* **Nearest Insertion**
* **Brute Force**

Additionally, the project aims to compare the efficiency of these algorithms when implemented in **Python** and **C++**, analyzing their performance in terms of execution time, memory usage, and language-specific optimizations.

## Problem Statement

### **Problem Statement:**

We aim to solve the **Traveling Salesman Problem (TSP)**, which is a fundamental problem in graph theory. It involves finding the shortest possible route that visits all vertices exactly once and returns to the starting point.  
Our goal is to:

1. Implement the TSP using the specified algorithms.
2. Compare the performance of the Python and C++ implementations.
3. Generate insights into the advantages and disadvantages of using these languages for graph problems.

## Code Implementation

1. **Programming Language 1:** Python (Using NetworkX and Tkinter for visualization)
   * **Graph generation:** A complete graph with random edge weights.
   * **GUI:** Tkinter for interactive interface, Matplotlib for graph plotting.
   * **Algorithm:** Nearest Neighbor to find an approximate Hamiltonian cycle.
   * **Visualization:** Highlight the cycle, display the starting vertex, and show the total weight.

2. **Programming Language 2:** C++

* + **Graph generation:** A complete graph with random edge weights.
  + **Algorithm:** Nearest Neighbor,Nearest insertion and Brute force to find an approximate Hamiltonian cycle.
  + **Visualization:**The program also calculates the total weight of the TSP cycle and measures execution time using the clock() function in C++.

## Efficiency Analysis

1. **Performance Metrics:**
   * **Execution time**: Measure the time taken by both Python and C++ implementations for various input sizes.
   * **Memory usage**: Analyze memory consumption in both versions.
   * **Usability comparison**: Document the ease of GUI implementation and any challenges faced with each language.
2. **Test Cases:**
   * Run the algorithm for graphs with 10, 20, and 30 vertices to observe performance and scalability.
   * Ensure both implementations produce the same correct output for all test cases.

## Expected Output

* A graph visualization highlighting the Hamiltonian cycle with labeled vertices and weights.
* Total path weight displayed on the GUI interface.
* Comparison tables of performance between Python and C++ implementations.

## Tools and Technologies

* **Python:** NetworkX, Matplotlib, Tkinter
* **C++:** ctime, algorithm, vectors
* **IDE:** Visual Studio Code for Python and C++