Project Proposal for Project For Applications of Graph Theory: Dijkstra's Algorithm

Project Title: Dijkstra's Algorithm in Python and C++

Group Members:

22K-4282 Daniyal Fahim
22K-4167 Daiyan Ur Rehman Khan
22K-4663 Dawood Adnan

Project Overview

The objective of this project is to explore, implement, and analyze Dijkstra's shortest path algorithm for graphs, comparing its efficiency in both Python and C++. Dijkstra's algorithm, widely used for finding the shortest path in weighted graphs, provides practical applications in network routing, spatial mapping, and navigation systems. By implementing the algorithm in two distinct languages, we aim to measure performance based on execution time and memory usage across multiple graph samples. Additionally, we will develop a Python-based GUI to visualize how vertices are connected, enabling better comprehension of the shortest path calculation process.

Objective

1. **Implementation**: Develop and document Dijkstra's algorithm in Python and C++ from scratch, utilizing each language's unique features to optimize the code.
2. **Efficiency Analysis**: Measure and compare both implementations' execution time and memory usage on a series of test graphs with 10+ vertices to assess each language's strengths and limitations.
3. **GUI Representation**: Design a GUI in Python for visualizing graph vertices, edges, and shortest paths to enhance understanding of the algorithm's application.

Problem Statement

The project seeks to answer the question: *How does the implementation of Dijkstra's algorithm vary in efficiency and performance between Python and C++?* This question will be addressed through practical coding, performance testing, and GUI development. We will analyze both languages' strengths in handling Dijkstra's algorithm, focusing on execution speed, memory management, and algorithm adaptability.

Methodology

1. **Algorithm Implementation :**
 - Python : Write a Python version of Dijkstra’s algorithm using efficient data structures such as dictionaries and priority queues to manage vertices and paths.
 - C++ : Implement the algorithm in C++, leveraging STL components (e.g., priority queues, vectors) to achieve comparable functionality while optimizing for speed.
-Create a non focused program on GUI of cpp and Python to ensure correct comparison and for GUI use python separately.

2. **Efficiency Testing and Analysis :**
 - Sample Graphs : Use more than 10 sample graphs of varying sizes and densities to observe performance in real-world conditions.
 - Performance Metrics : Measure execution time.
 - Comparative Analysis : Assess the strengths and weaknesses of Python and C++ for this problem based on collected metrics, noting how each language's features affect algorithmic efficiency.

3. **Graphical User Interface (GUI)** :
 - Develop a Python-based GUI to visualize graph vertices, edges, and shortest path calculations, showcasing Dijkstra’s algorithm in action.
 - Include user options to input vertices and edges, with graphical representation updates based on the calculated shortest paths.

Deliverables

1. Code Implementation :

2. Performance Analysis Report :

3. GUI Representation :

4. Presentation :

AIM
By completing this project, we aim to deepen our understanding of graph theory applications, comparing the efficiency of Python and C++ in solving shortest path problems and effectively demonstrating this through both code and visualization.