Project 3: Algorithms

Ayobami Ajala R11720649

Leslie Omonzane R11756088

Eworitse Mabuyaku R11710212

Oluwaferanmi Ogunniran R11760605

Bellam Ford Algorithm

**Interpretation of the Given Problem**

* The task entails identifying the most efficient routes between nodes in a graph by applying the Bellman-Ford algorithm. In this context, nodes correspond to various locations within a college campus, and the edges signify the distances between these locations.
* The importance of addressing this challenge is elaborated upon, underscoring its pertinence in enhancing campus navigation and optimizing resource allocation. Special attention is dedicated to any distinctive constraints or requirements inherent to the campus environment that impact the interpretation of the problem.

**Methodology of the Solution**

**Graph Class Initialization:**

* Setting Up the Graph Class: We start by initializing the Graph class, carefully incorporating any campus-specific details.
* Campus Parameters: Specific campus-related parameters are integrated into the initialization process.
* Algorithm Selection: We chose the Bellman-Ford algorithm for its suitability in handling scenarios with potentially negative-weight edges.

**Bellman-Ford Algorithm:**

* Implementation Complexities: The implementation within the Graph class involves intricate steps.
* Iteration Choices: We explain the decision to iterate V-1 times, a crucial aspect of the algorithm.
* Detecting Negative Cycles: Clear insights into how the algorithm detects negative cycles are provided.

**Experimental Results**

**Graph Creation:**

* Process Overview: We provide a detailed walkthrough of the graph creation process, illustrating how it represents various campus locations.

**Application of Bellman-Ford Algorithm:**

* Scenario Exploration: We extend the application of the Bellman-Ford algorithm to multiple scenarios. This includes considering different source nodes and experimenting with varying edge weights.

**Result Presentation:**

* Path Visualization: Apart from visually representing paths, we explore additional avenues for presenting results.
* Statistical Metrics: We delve into the use of statistical metrics to quantify and analyze the algorithm's performance.

**Conclusions**

**Key Findings:**

* The Bellman-Ford algorithm effectively determines optimal paths between campus locations, providing valuable insights into efficient navigation.
* Varied scenarios, including different source nodes and edge weights, were explored, showcasing the algorithm's adaptability.
* Statistical metrics and graphical representations offer a comprehensive understanding of algorithm performance.

**Team Collaboration:**

* **Working Together:** The team collaborated to solve the problem, with each member bringing their skills to the project.
* **Clear Roles:** Everyone had specific roles and responsibilities, ensuring a focused approach.

**Communication Strategies:**

* **Regular Updates:** The team had regular updates to keep everyone in the loop about progress.
* **Open Communication:** A culture of open communication allows for the free exchange of ideas and addressing concerns.

**DIJKSTRA’S ALGORITHM**

**Interpretation of the Given Problem:**

1. **Real-World Mapping:**
   * We translated a real-world scenario, such as a university campus, into a computational model. The graph elements accurately represented the locations, paths, and distances on the campus.
2. **Weighted Graph Significance:**
   * The inclusion of weights on edges was crucial, capturing not only connectivity but also the varying complexities of paths. This nuanced approach enhanced our understanding of the campus layout.
3. **Campus Mapping:**
   * Our code implemented Dijkstra's algorithm, creating a model of a campus map. Nodes corresponded to locations, edges to paths, and distances to weights or lengths. The primary goal was to find the shortest path from a specified starting point to all other locations.
4. **Shortest Path Objective:**
   * We actively worked to efficiently find the shortest paths from a start point to all locations on the campus. Dijkstra's algorithm served as our chosen method, addressing this classic problem in graph theory.

**Methodology of the Solution:**

**Dynamic Graph Evolution:**

* The graph's dynamic nature enables an evolving representation of the campus, accommodating changes in layout or the addition of new locations, ensuring adaptability to future developments.

**Graph Integrity Maintenance:**

* The implementation ensures the integrity of the campus map model, guaranteeing distinct nodes, correctly linked edges, and accurately associated distances, providing a reliable representation of the campus.

**Graph Initialization:**

* The Graph class is defined to represent the campus map.
* Attributes include nodes (Node), edges (Edge), and distances (Dist).
* Node is a set to store unique nodes.
* Edge is a defaultdict to store edges between nodes.
* Dist is a dictionary to store distances between nodes.

**Dynamic Node and Edge Addition:**

* The solution methodology incorporates the flexibility to dynamically add nodes and edges to the graph. This feature is crucial for adapting the campus map representation to changing requirements or evolving campus structures**.**

**Experimental Results:**

**Versatile Campus Modeling:**

* The code effectively represents campus maps, adaptable to different layouts and structures.

**Robust Algorithm:**

* Dijkstra's algorithm consistently finds optimal paths in the dynamic campus map, ensuring reliability.

**Successful Graph Modeling:**

* The code accurately model’s campus connections with nodes, edges, and distances.

**Accurate Pathfinding:**

* Dijkstra's algorithm navigates the campus map efficiently, demonstrating pathfinding accuracy in experimental results.

**Top of Form**

**Conclusions:**

**Navigational Efficiency:**

* Conclusive results emphasize the algorithm's efficiency in navigating the campus, finding shortest paths for optimized movement and potentially enhancing overall accessibility.

**Insights into Campus Connectivity:**

* Information about visited nodes and gateways goes beyond pathfinding, providing valuable insights into the broader connectivity patterns within the campus.

**Team Considerations:**

**Scalability Assurance:**

* Our modular design and clear variable naming ensure scalability, allowing the codebase to handle additional features or complexities as our team expands or the project evolves.

**Knowledge Transfer Facilitation:**

* The comprehensibility of our code, supported by documentation and comments, eases knowledge transfer within our team. Any new members can quickly understand the logic and contribute effectively.