Oghenetega Emagha 19041139

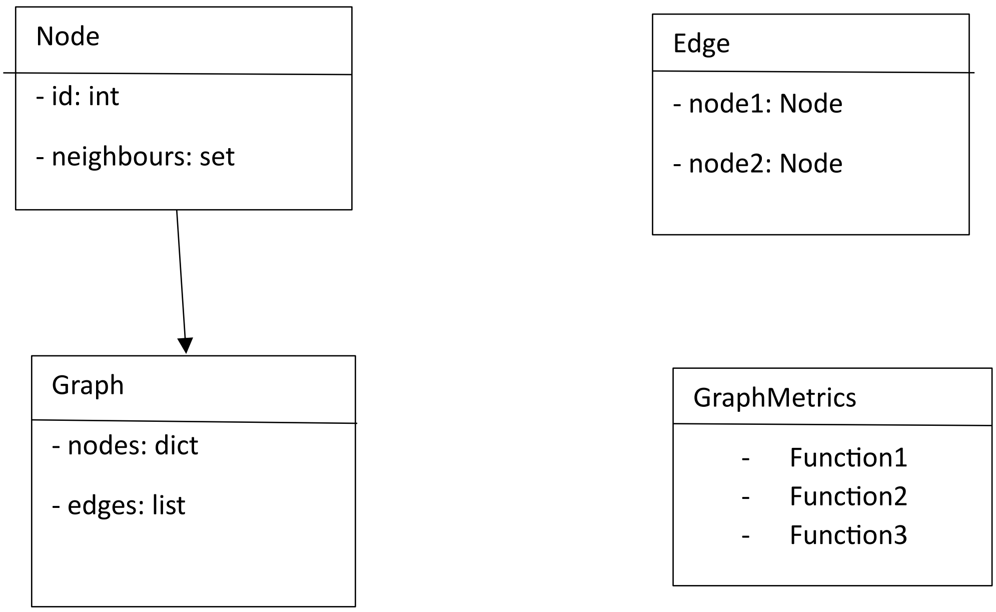
5COM2003 Practical Assignment Report

World Design

The world is modelled as an undirected graph with seven vertices and edges. The implementation employs a structured approach with dedicated classes for Node, Edge, and Graph.

Graph Implementation: The design choice prioritizes clarity and modularity. Each class encapsulates its attributes and functionalities, ensuring a clear separation of concerns. Nodes represent individual entities with unique identifiers and adjacency information, while edges denote connections between nodes. The Graph class orchestrates the management of nodes and edges, providing a centralized interface for graph operations.

UML Class Diagram:



Design Choices Explanation: The design choices for the UML class diagram are aimed at clarity and modularity. Each class encapsulates its properties and behaviours: ‘Node’ represents individual nodes with unique identifiers and neighbour relationships, ‘Edge’ signifies connections between nodes, and ‘Graph’ manages nodes and edges. The diagram illustrates their relationships, showcasing how ‘GraphMetrics’ could interact with Graph for metric computation, ensuring a comprehensive representation of the graph structure and operations.

World Metrics

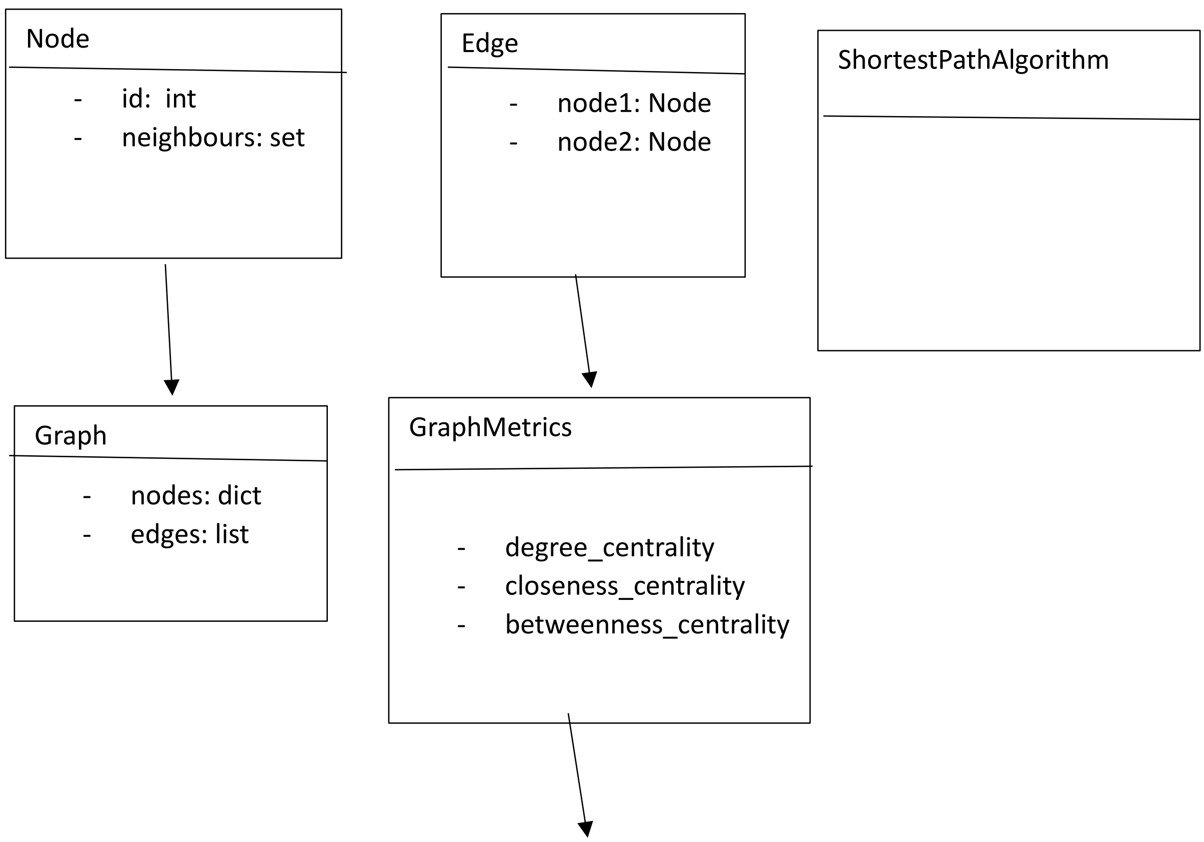
The implementation includes a GraphMetrics class with functionalities for computing three graph metrics.

Design Choice Explanation: For computing shortest paths (geodesics) in the Betweenness Centrality function, I utilized a well-known algorithm such as Dijkstra's algorithm due to its efficiency and accuracy in finding the shortest path between nodes. By pre-computing and storing these paths, we avoid redundant computations during metric calculations, enhancing performance and scalability.

Agent Design

The agent can navigate from a start node to a target node using two modes, Random Walk and Shortest Path. The agent retains the memory of visited nodes during each episode, enabling subsequent analyses.

UML Class Diagram:



Agent

- graph: Graph
- shortest_paths: dict
- episode_memory: list

random_walk()
shortest_path()
get_episode_memory()
sense_current_state()

Simulation

A total of 2000 simulations (1000 each for Random Walk and Shortest Path) were conducted. Each simulation involved randomly selection start and target nodes, counting visited nodes, and storing results.

Evaluation

Movement Mode	Average Nodes Visited
Random Walk	10.112
Shortest Path	3.033

Analyses of the obtained results:

Differences: The random walk mode results in visiting more nodes on average compared to the shortest path mode.

Reasons: Random walk explores the graph without a predetermined path, leading to more nodes being traversed. Shortest path mode focuses on efficiency by following the most direct route.

Surprises: While the disparity in average visited nodes aligns with expectations, the precise numerical difference may be unexpected, highlighting the impact of different navigation strategies on traversal patterns.

Conclusion

This report reflects a meticulous approach to designing and evaluating the agent's navigation strategies within the given graph environment. The combination of thoughtful design choices and rigorous simulations provides valuable insights into the efficiency and effectiveness of different movement modes.