TEB1113/TFB2023 Algorithm and Data Structure

January 2025 Trimester

Mini Project

Title: Implementation of Dijkstra's Algorithm

This mini project requires collaborative teams of two to four students. Each team will implement Dijkstra's Algorithm utilizing a priority queue data structure. A comprehensive report must be submitted, detailing the implementation process, sample executions, and a thorough discussion. The report should be structured as follows:

1. **Introduction**:
    - Overview of Dijkstra's Algorithm and its significance.
    - Explanation of priority queues and their role in optimizing the algorithm.
2. **Methodology**:
    - Detailed description of the algorithm's implementation steps.
    - Justification for choosing specific data structures and programming approaches.
3. **Implementation**:
    - Presentation of the code with explanations.
    - Discussion on how the priority queue was integrated into the algorithm.
4. **Sample Runs**:
    - Illustration of test cases with corresponding outputs.
    - Analysis of results to verify the correctness and efficiency of the implementation.
5. **Discussion**:
    - Reflection on challenges faced during implementation and how they were addressed.
    - Comparison of the implemented algorithm's performance with theoretical expectations.
6. **Conclusion**:
    - Summary of findings and overall learning outcomes from the project.
7. **References**:
    - List of resources and literature consulted during the project.

This structure ensures a comprehensive documentation of the project, highlighting both the technical and analytical aspects of the work.

Objective

- to instil a team spirit in completing a programming project
- to understand and implement Dijkstra's Algorithm, which identifies the shortest paths from a starting vertex to all other vertices in a weighted graph with non-negative edge weights.
- To implement the Dijkstra's Algorithm.

Background

Dijkstra's Algorithm is a fundamental algorithm used to find the shortest path between nodes in a graph.

Algorithm Overview

Dijkstra's Algorithm operates by keeping track of unvisited nodes and assigning each a tentative distance: zero for the starting node and infinity for all others. The algorithm shall repeatedly

- select the unvisited node with the smallest tentative distance,
- examine the unvisited neighbours,
- calculate their tentative distances through the current node, and
- update these distances if they are smaller than previously recorded values.

This process continues until all nodes have been visited.

Procedure

1. **Graph Representation**:
    - Graphs can be represented using various methods, notably *adjacency lists* and *adjacency matrices*. In this lab, we'll utilise an *adjacency list* representation, where each node maintains a list of its neighbouring nodes along with the corresponding edge weights.
2. **Implement the Dijkstra's Algorithm**:
    - Initialize the graph as an *adjacency list*.
    - Create a *priority queue* to keep track of the next node to process based on the shortest known distance.
    - Initialize a dictionary to store the shortest path to each node.
    - While there exists node(s) to process in the priority queue:
        - ❖ Extract the node with the smallest tentative distance.
        - ❖ For each unvisited neighbour, calculate the tentative distance through the current node.
        - ❖ If this distance is less than the known distance, update the shortest path and add the neighbour to the priority queue.

3. **Tasks**:

    - Implement the Dijkstra's Algorithm using a priority queue as the data structure.
    - Test the Implementation by define a sample graph as an adjacency list:

```
graph = {
   'A': {'B': 2, 'C': 6, 'F': 3},
   'B': {'A': 2, 'C': 4, 'D': 7, 'E': 5},
   'C': {'A': 6, 'B': 4, 'D': 3, 'G': 1},
   'D': {'B': 7, 'C': 3, 'E': 1, 'G': 2},
   'E': {'B': 5, 'D': 1},
   'F': {'A': 3},
   'G': {'C': 1, 'D': 2}
}
```

    Run the algorithm with 'A' being the start node before determining the shortest paths. Finally, display the output of the shortest paths.

    In your report, in addition to the code and sample runs, include a discussion on time complexity of Dijkstra's Algorithm using priority queue as well as the pros and cons of Dijkstra's Algorithm over Bellman-Ford Algorithm.

    Submit your report **by 10 AM on 20th March 2025**.

References

en.wikipedia.org

geeksforgeeks.org