CMPE 224/343

Spring 2024

Programming Homework 2 Report

1. Information

- ID: 44293566706
- Name: Erkan Sancak
- Section: 2
- Assignment Number: 2

2. Problem Statement and Code Design

- **Problem Summary:** For Homework 2, the objective is to implement solutions for two distinct problems using graph data structures. Problem 1 involves creating a directed graph and finding two-step paths from a given starting point. Problem 2 requires reading graph data from a text file, identifying non-connected neurons, and printing relevant graph information.

  Code Design: For each problem, the code has been structured into modular components for clarity and organization.

  Problem 1: Finding Two-Step Paths

- **DirectGraph class:** Represents the directed graph using adjacency lists. Provides methods for graph manipulation. DirectGraph encapsulates the graph data structure and offers methods for adding edges and finding two-step paths
- **FileRead class:** Reads graph data from a text file and constructs the graph. FileRead reads graph data from a text file and constructs the graph.
- **HW2_Solution class:** Main program that handles file reading and graph operations. HW2_Solution serves as the entry point, managing file reading and graph operations
- **ValueFinder class:** Utility class for converting string values to integers. ValueFinder provides utility methods for converting string values to integers.

  Problem 2: Identifying Non-Connected Neurons

- **FileRead class:** Reads graph data from a text file and constructs an adjacency list. FileRead reads graph data from a text file and constructs the graph represented as an adjacency list.
- **HW2_Q2_Solution class:** Main program that reads graph data, identifies non-connected neurons, and prints graph information. HW2_Q2_Solution orchestrates file reading, graph processing, and result printing.

- **ValueFinder class:** Utility class for converting string values to integers. ValueFinder provides utility methods for converting string values to integers.

3. Implementation and Functionality

Problem 1: Finding Two-Step Paths

Sub-module 1: DirectGraph Class

- **Functionality:**
  - Represents a directed graph using adjacency lists.
  - Provides methods for adding edges and finding two-step paths from a given vertex.

Sub-module 2: FileRead Class

- **Functionality:**
  - Reads graph data from a text file and constructs the graph.

Sub-module 3: HW2_Solution Class

- **Functionality:**
  - Acts as the main program, orchestrating file reading and graph operations.

Sub-module 4: ValueFinder Class

- **Functionality:**
  - Provides utility methods for converting string values to integers.

Problem 2: Identifying Non-Connected Neurons

Sub-module 1: FileRead Class

- **Functionality:**
  - Reads graph data from a text file and constructs an adjacency list.

Sub-module 2: HW2_Q2_Solution Class

- **Functionality:**
  - Main program that reads graph data, identifies non-connected neurons, and prints graph information.

Sub-module 3: ValueFinder Class

- **Functionality:**
  - Provides utility methods for converting string values to integers.

4. Testing

- **Problem 1 (Finding Two-Step Paths):**
    - The solution effectively identified all two-step paths in various graph structures.
    - Performance remained satisfactory even for graphs with a large number of vertices and edges.
    - Error handling mechanisms adequately handled invalid inputs and edge cases.
- **Problem 2 (Identifying Non-Connected Neurons):**
    - The solution accurately identified non-connected neurons in diverse graph configurations.
    - Execution time remained reasonable across different graph sizes and connectivity patterns.
    - The program robustly handled cases where all neurons were interconnected or when no non-connected neurons were present.

5. Final Assessments

Achievements

- Successfully implemented a solution to find all two-step paths in a directed graph.
- The solution demonstrated robustness and accuracy across various input scenarios.
- Modular design facilitated easy integration and testing of individual components.
- Developed an effective solution to identify non-connected neurons in a directed graph.
- The solution exhibited reliability and accuracy in detecting non-connected components across various graph configurations.

Challenges

- Ensuring the efficiency of the algorithm for large graphs with numerous vertices and edges required careful optimization.
- Error handling mechanisms needed to handle unexpected edge cases to prevent runtime errors.