Written Report

A. Project Overview

Goal:
To analyze New Jersey train performance using delay data and identify central stations and problematic routes based on delay metrics.

Dataset:
Source: Kaggle (NJ Transit performance data)
Size: 50,000+ records in CSV format (local path: `src/stations_filtered.csv`)
Link omitted as file exceeds GitHub storage limits.

B. Data Processing

Loading:
CSV data is loaded using Rust's `csv::ReaderBuilder` and deserialized into a custom `TrainRecord` struct using `serde`.

Cleaning & Transformations:

- Amtrak records are filtered out. (Python)
- Rows missing delay_minutes, from, or to are dropped. (Python)
- Whitespace is stripped from from and to station names. (Python)
- Randomly sampled data with 1001 data points is assigned to stations_filtered.csv.(Python)
- Cleaned records are written to stations_filtered.csv. (Python)
- Only edges with valid delays are used to construct the graph.
- Graph edges are stored as (from_station, to_station, delay) tuples.

C. Code Structure

Modules:

- load.rs: Loads and deserializes the dataset.
- graph.rs: Defines the transit graph structure and builds it from the records.
- metrics.rs: Contains algorithms to compute graph metrics like shortest paths, closeness, betweenness, and delay ranking.
- main.rs: loads data, builds the graph, computes metrics, prints results, and tests metrics

Key Types & Functions:

- TrainRecord: Struct representing each row in the dataset.
- Purpose: Model train trips and delays as structured data.
- Input: CSV file with columns like from, to, delay_minutes.
- Output: Rust struct with typed fields for each column.

- TransitGraph (Struct):
- Purpose: Represent the train network as a directed graph where nodes are stations and edges are weighted by delay.
- Input: Vector of TrainRecords.
- Output: HashMap of stations to vectors of destination-delay pairs.

- from_records():
- Purpose: Construct a TransitGraph from train data.
- Input: Slice of TrainRecords.
- Output: Initialized TransitGraph.
- Logic: Iterate over records, filter for valid delays, and populate a node-edge mapping.

- shortest_path():
- Purpose: Compute shortest path (in delay time) between two stations.
- Input: Start and end station names.
- Output: Option tuple of total delay and path vector.
- Logic: Implements Dijkstra's algorithm with a binary heap (priority queue) for efficiency.

- closeness_centrality():
- Purpose: Compute how central a station is based on its accessibility (lower total delay to others).
- Input: A station name.
- Output: Option float centrality score.
- Logic: Sums shortest path delays to all reachable stations, then returns reachable count divided by that sum.

- betweenness_centrality():
- Purpose: Compute how often each station appears in shortest paths between others.
- Input: None explicitly (runs for all nodes).
- Output: HashMap from station to centrality score
- Logic: Implements Brandes' algorithm using BFS and dependency accumulation.

- get_route_average_delays():
- Purpose: Calculate average delay for each route between stations.
- Input: None explicitly.
- Output: Vector of ((from, to), average_delay, count).
- Logic: Aggregate total delay and trip counts by (from, to) key, then compute average.

Main Workflow:

1. Load CSV records into `TrainRecord` structs.
2. Use `from_records` to build `TransitGraph`.
3. Run centrality and delay functions.
4. Print top stations/routes by ranking.

D. Tests

```
running 5 tests
test test_closeness_is_finite_for_main_station ... ok
test test_load_real_data ... ok
test test_rank_routes_by_average_delay_sorted_descending ... ok
test test_real_shortest_path_exists ... ok
test test_betweenness_non_negative ... ok

test result: ok. 5 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 22.47s
```

Test Descriptions and Importance:

test_load_real_data:
Purpose: Confirms that the data loading pipeline is functional and that the dataset is large enough to analyze.
Why it matters: Ensures the project starts with a valid and substantial dataset, verifying basic I/O and filtering logic.

test_real_shortest_path_exists:
Purpose: Checks that a valid shortest path exists between two major stations.
Why it matters: Validates the correctness of the graph construction and pathfinding logic in realistic scenarios.

test_closeness_is_finite_for_main_station:
Purpose: Ensures that a well-connected station returns a usable, finite closeness centrality score.
Why it matters: Verifies that the centrality logic is robust and avoids divide-by-zero or unreachable node errors.

test_betweenness_non_negative:
Purpose: Checks that betweenness scores are never negative.
Why it matters: Centrality scores should represent counts of paths and influence, which must be non-negative, this ensures algorithm correctness.

test_rank_routes_by_average_delay_sorted_descending:
Purpose: Verifies that route delays are correctly sorted in descending order.
Why it matters: Ensures the ranking output is reliable for identifying worst-performing routes and making data-driven decisions

E. Results

Program Output:

```
Top 10 stations by closeness centrality:
 1. Montclair Heights              8.5714
 2. Mount Arlington                5.4545
 3. Basking Ridge                  3.7113
 4. Emerson                        3.1579
 5. Westwood                       1.1538
 6. Hackettstown                   0.9091
 7. Lebanon                        0.6667
 8. Point Pleasant Beach           0.6383
 9. Bernardsville                  0.5463
10. White House                    0.5263
Top 10 stations (unweighted betweenness):
 1. Newark Broad Street            7791.9707
 2. Newark Penn Station            5945.4692
 3. Secaucus Lower Lvl             5264.1660
 4. Secaucus Upper Lvl             4775.9976
 5. Hoboken                        4746.0000
 6. Summit                         2827.8718
 7. Newark Airport                 1993.7648
 8. Westfield                      1927.0001
 9. Rahway                         1825.1406
10. Woodbridge                     1814.0001
Top 10 routes by average delay:
 1. Metropark → Rahway : 13.90 minutes (7 trips)
 2. New Brunswick → Jersey Avenue : 12.03 minutes (6 trips)
 3. Orange → Brick Church : 9.29 minutes (5 trips)
 4. Newark Airport → Newark Penn Station : 9.03 minutes (6 trips)
 5. Delawanna → Lyndhurst : 7.10 minutes (5 trips)
 6. North Elizabeth → Elizabeth : 6.74 minutes (10 trips)
 7. Short Hills → Summit : 6.06 minutes (6 trips)
 8. Ramsey Route 17 → Mahwah : 5.65 minutes (5 trips)
 9. Metropark → Metuchen : 5.51 minutes (5 trips)
10. Maplewood → South Orange : 5.38 minutes (8 trips)
Top 10 routes by **lowest** average delay:
 1. Long Branch → Long Branch : 0.00 minutes (5 trips)
 2. Newark Broad Street → Hoboken : 0.00 minutes (5 trips)
 3. Princeton → Princeton : 0.15 minutes (5 trips)
 4. Secaucus Lower Lvl → Hoboken : 0.20 minutes (10 trips)
 5. Edison → Metuchen : 1.25 minutes (10 trips)
 6. Ramsey Main St → Allendale : 1.29 minutes (5 trips)
 7. Hamilton → Princeton Junction : 1.30 minutes (7 trips)
 8. Glen Rock Main Line → Hawthorne : 1.53 minutes (5 trips)
 9. New York Penn Station → Secaucus Upper Lvl : 1.62 minutes (10 trips)
10. Hoboken → Hoboken : 1.76 minutes (14 trips)
```

- Closeness Centrality:
- Montclair Heights ranks highest, indicating it has fast access (in terms of delay) to many other stations, making it a highly connected node in this dataset. Other stations like

Mount Arlington and Basking Ridge also stand out, likely due to their strong positioning in local service networks with low overall delays to others.

- Betweenness Centrality:
- Newark Broad Street, Newark Penn Station, and both levels of Secaucus rank highest,showing they act as major intermediaries in the network. These stations lie on many of the shortest paths between other pairs, underscoring their importance in routing and transfer efficiency.
- Routes with the Highest Average Delay(≥5 trips):
- Routes such as Metropark → Rahway or New Brunswick → Jersey Ave offer insight into where mid-level congestion still accumulates.
- Routes with the Lowest Average Delay (≥5 trips):
- Routes like Newark Broad Street → Hoboken and Princeton → Princeton show extremely low average delays, indicating either strong reliability or short route distance. These entries all meet the minimum trip count threshold (5+ trips), adding more statistical confidence to their ranking.

- Overall Implication:
- These metrics can guide service planning by spotlighting:
- Which stations hold structural or strategic importance (e.g., high betweenness),
- Which areas experience performance gaps, and
- Where consistent on-time performance is achievable.
- The inclusion of a trip count threshold helps avoid misleading conclusions from low-sample routes.

F. Usage Instructions

Build:
Run `cargo build` in the project root.

Run:
Use `cargo run`. Output is printed to terminal.

Tests:
Run `cargo test`

Estimated Runtime:
5 seconds