Morgan Walsh

Write Up for Final Project

DS 210

4/30/25

Boston Crash Data Set Analysis

***Goal**: Identify Boston intersections with the most crash activity and visualize their frequency and severity patterns via histograms.*

Dataset: MassDOT Open Data Portal for Car Accidents. The range selected is the past 5 years, so 2021 to present. It has 22454 entries. Columns used include crash number, injuries, intersection, and coordinates.

Data Processing

- Loaded with csv::Reader and deserialized into CrashRecord.
- Transformed with .filter_map() into ProcessedCrashRecord:
    - Parsed date strings
    - Parsed time strings
    - Filtered rows missing coordinates

Modules

- **data_loader**: Reads CSV and returns a vector of the processed data.
- **data_structures**: Contains structs of CrashData and ProcessedCrashData and the from_raw() constructor.
    - CrashRecord represents one row in crash_data.csv
    - ProcessedCrashData is cleaned and parse version of CrashRecord
    - ProcessedCrashRecord converts a raw crash record to a processed format
    -
- **analysis**:
    - group_by_intersections(): bins spatial data
    - build_crash_graph(): creates adjacency list
    - compute_degree_distribution(): maps node id to degree
    - top_n_high_degree_nodes(): identifies most connected intersections
    - is_severe(): filters fatal or injury crashes
- **visualization**:
    - plot_degree_histogram(): renders a PNG bar chart

Main Workflow

1. Load and clean CSV data
2. Group crashes into clusters binned intersections
3. Create undirected graph based on proximity
4. Compute degree distribution
5. Plot histogram of node degrees
6. Print top 5 intersections by degree
7. 2–6 is repeated using only severe crashes rated by injuries

TESTS

```
    Finished `test` profile [unoptimized + debuginfo] target(s) in 1.79s
     Running unittests src/main.rs (target/debug/deps/intersections-8a3bfe6fa514e973)

running 2 tests
test tests::test_grouping ... ok
test tests::test_struct_population ... ok

test result: ok. 2 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.00s

[/opt/app-root/src/Final_Project_210/boston_crashes/intersections]
```

Test_struct_population: it checks to make sure ProcessedCrashRecord is using the function from_raw properly. It creates a sample crash record to check that strings are parsed correctly and coordinates are being preserved, which is the backbone of further analysis.

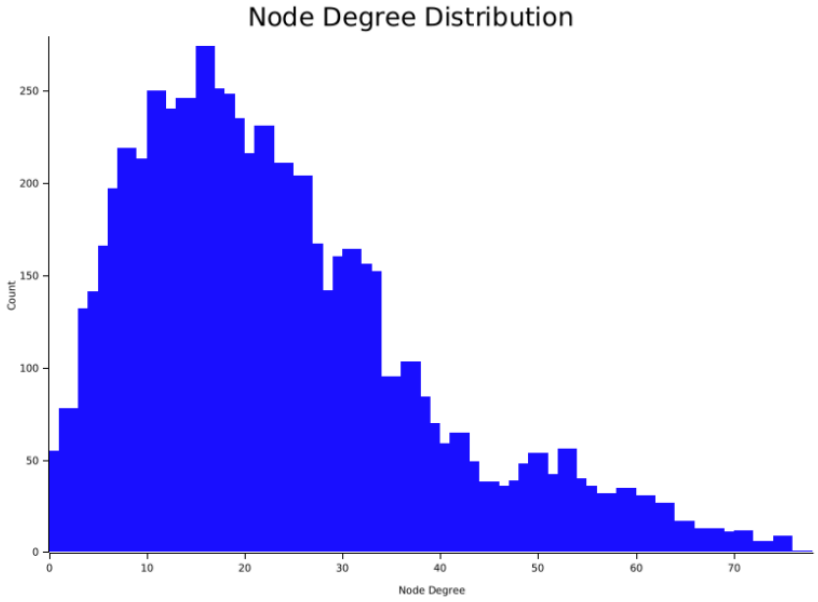
Test_grouping: verifies that spatial clustering and graph building is working as expected. The two mock ProcessedCrashRecords should merge within their distance to each other so this makes sure clusters that should be added together are. Since there is only one graph node, that's what it needs to check equality for in the graph.

RESULTS

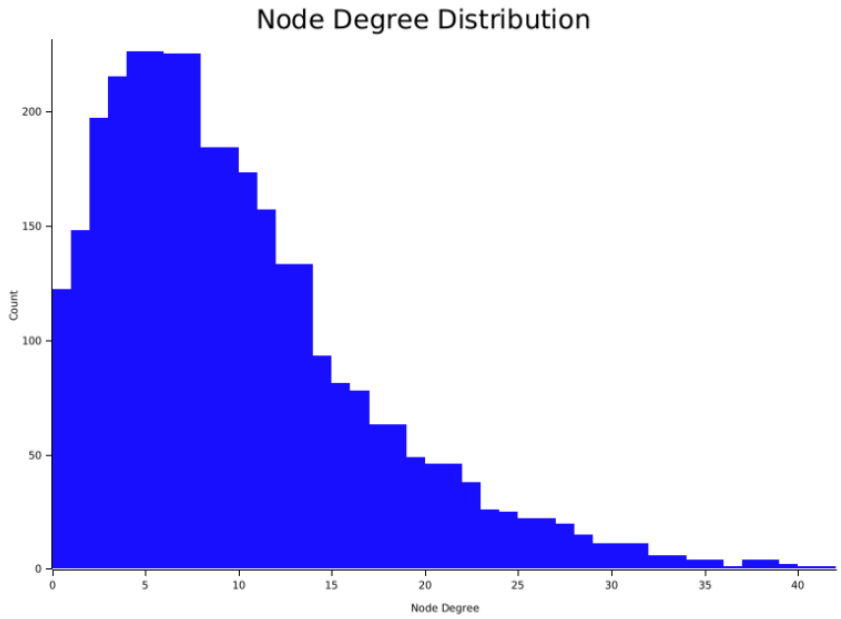
Terminal Output:

```
Loaded 20519 crash records.
Built graph with 7801 intersections & 90881 edges.
Computed degrees for 7755 nodes
Histogram saved to histogram_output/degree_histogram.png
Top 5 highest-degree intersections:
charlesgate east rte  / marlborough st rte (Degree: 77) at approx. coords (9346.00, 36000.00)
Intersection at (9440.00, 36078.00) (Degree: 76) at approx. coords (9440.00, 36078.00)
charlesgate east / beacon street (Degree: 75) at approx. coords (9346.00, 36002.00)
Intersection at (9439.00, 36077.00) (Degree: 75) at approx. coords (9439.00, 36077.00)
Intersection at (9343.00, 36000.00) (Degree: 75) at approx. coords (9343.00, 36000.00)
Filtered to 5474 severe crashes
Top 5 intersections with most severe crashes:
1.  (Severe crashes: 43) at approx. coords (9465.00, 35887.00)
2.  (Severe crashes: 37) at approx. coords (9478.00, 36038.00)
3.  (Severe crashes: 27) at approx. coords (9474.00, 36046.00)
4.  (Severe crashes: 26) at approx. coords (9505.00, 35709.00)
5.  (Severe crashes: 23) at approx. coords (9432.00, 35690.00)
Histogram saved to histogram_output/severe_crash_degree_histogram.png
Duration 2.53s
```

Degree Histogram PNG:



PNG of Crash Severity Histogram:



Findings: Few nodes had very high connectivity, which means that very few areas in Boston have high amounts of crashes or crashes that are severe. The top 5 intersections with high connectivity are likely busy or confusing intersections that become crash hot spots in the city. Top 5 crash severity clusters may have poor traffic planning or high speed limits compared with the rest of Boston.

Usage Instructions

This program takes data/crash_data.csv as input. If one wanted to use a different data set, it could be uploaded to the data file, and file_path in main should be changed to the csv title. Histograms output by this program are stored as pngs in histogram_output.

- Cargo run will return the generated histograms and a ranking of the 5 highest intersection clusters with crashes and then a second top 5 ranking of the clusters with the most severe crashes.
- Print statements used for debugging have been commented out but can be reinserted. Cargo test will run test modules for struct propagation and cluster logic.

SOURCING:

I got help from chat gpt when debugging my histograms png as we hadn't covered it in class. I read the git hub plotters documentation when first building my histogram generation function -> https://plotters-rs.github.io/book/basic/basic_data_plotting.html

When making my clusters, I had lightly researched other forms of analyzing crash data and found this paper talking about using euclidean distance to measure the distance between the

clusters and I found it work well (I think at least) in my function. Link:
https://www.nature.com/articles/s41598-024-81121-7