**Group 56**
Anhelina Liashynska (Github: angellsh)
William Johnson (Github: johnsonwilliam3)
Daniel Lopez (Github: daniel-plops)
**Project Repo:** https://github.com/johnsonwilliam3/Project-3
**Video Demo:** https://youtu.be/aItlPmedtZQ

**What problem are we trying to solve?**

Our goal is to determine critical locations (cities) for prioritizing new road construction across the United States. Initially, this prioritization will focus on congestion and centrality ranks due to time constraints and the need for accuracy and efficiency. By focusing on congestion and centrality, we aim to enhance road efficiency, reduce congestion, and support economic and social development.

**Why is this a problem?**

Efficient prioritization of road construction is crucial for managing traffic congestion, improving road safety, and fostering economic growth. Misallocation of resources can lead to higher costs, increased traffic delays, and missed opportunities for community development. Addressing this issue through a data-driven approach will provide significant benefits in terms of economic efficiency, quality of life, and infrastructural planning.

**When do we know that we have solved the problem?**

- The application calculates a “Congestion Index” for each city to prioritize enhancements.
- Cities are ranked based on their priority for new road projects, which is based on the congestion severity and centrality rank.
- Results are visualized in a clear and interpretable format.
- The solution is validated with real-world data and provides actionable insights.

**Public Datasets Used:**

1. **US Traffic Congestions (2016-2022) -** Dataset Link (Sample Data)

Contains 2 million traffic events with congestion severity ranks calculated from the duration and distance of the congestion.

Reference: Moosavi, Sobhan, Mohammad Hossein Samavatian, Arnab Nandi, Srinivasan Parthasarathy, and Rajiv Ramnath. "Short and Long-Term Pattern Discovery Over Large-Scale Geo-Spatiotemporal Data." In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2905-2913, 2019.

2. **US Zip Codes Database -** Dataset Link

Used for mapping and calculating zip codes for destination cities in the graph.

**Tools/Languages/APIs/Libraries Used:**

1. **C++**: Core application development and algorithm implementation.
2. **CSV/JSON Libraries**: For data parsing and handling.
3. **Folium**: For displaying maps and visualizing geographical data.
4. **Pandas**: For data manipulation, analysis, and visualization.
5. **Matplotlib/Seaborn**: For creating visualizations and charts for testing.

**Algorithms Implemented:**

- **Betweenness Centrality.** Brandes' Betweenness Centrality algorithm helps in identifying cities or intersections that serve as crucial connections within the network. Nodes with high betweenness centrality are those that frequently lie on the shortest paths between other nodes. This means they act as critical intermediaries in the network. If these nodes experience congestion, it can have widespread effects on overall traffic flow. Freeman (1977) demonstrated that nodes with high betweenness centrality are key to network connectivity and can influence the overall efficiency and robustness of the network.
- **Degree Calculation:** To find the degree of nodes (i.e., the number of connections a node has), we use a simple traversal of the adjacency list. This involves counting the number of edges incident to each node.
- **Priority Queue:** Used to manage and rank regions based on the Ideal Index, using a data structure that allows efficient insertion and extraction of elements based on their priority.

**Additional Data Structures/Algorithms Used:**

- **Graph Representation:** Adjacency lists are used to represent the road network, facilitating efficient edge insertion and traversal operations.
- **Heuristic Approximations:** To manage the computational load of betweenness centrality calculations, heuristic methods are employed. Heuristic approximations target nodes that are likely to have the greatest impact on network performance. Nodes with high congestion and degree are often central to the network's traffic flow and congestion issues. By prioritizing these nodes, the heuristic approach ensures that the analysis focuses on areas most likely to benefit from optimization, making the problem manageable within reasonable time constraints.

**Betweenness Centrality vs. Degree Calculation:**

Betweenness centrality and degree calculation are comparable metrics for assessing node importance within a network, though they approach this task differently. Betweenness centrality provides a deeper, more nuanced understanding of a node's role by evaluating how frequently it lies on the shortest paths between other nodes, highlighting its importance in network connectivity and flow. In contrast, degree calculation offers a more straightforward measure of node importance by counting the number of direct connections (edges) a node has, providing a simpler but less comprehensive view of its connectivity.

**Distribution of Responsibility and Roles**

Project Structure and Organization: Anhelina Liashynska, William Johnson

Designing Algorithms:

- Loading data into a graph: Anhelina Liashynska
- Creating the graph structure: William Johnson, Anhelina Liashynska
- Creating the queue: William Johnson, Daniel Lopez
- Calculating indexes for each city: Anhelina Liashynska
- Graph traversal: Anhelina Liashynska, Daniel Lopez

Other:

- Displaying a colored map based on index: William Johnson
- Data Collection, Preprocessing and Merging: Anhelina Liashynska
- Designing Interface: William Johnson
- Interactive Map Design: William Johnson
- Written Documentation: Anhelina Liashynska, Daniel Lopez
- Video Submission: William Johnson

Big O Worst Case Time Complexity Analysis

- Betweenness Centrality Calculation: O(n3)O(n^3)O(n3) using Brandes' algorithm for 200,000 data points, though heuristics reduce this complexity significantly.
- findMean and findSTD: O(n) since each algorithm goes through each data point a linear amount of times.
- filterOutNodesBasedOnRank: O(n) using a linear traversal discriminating nodes by rank.
- filterOutNodesBasedOnDegree: O(E) where E is the number of edges between the data points representing cities and each edge is used in the algorithm.
- findImportantNodes: O(n + E) since the algorithm utilizes filterOutNodesBasedOnRank and filterOutNodesBasedOnDegree to return the most important nodes.
- findFinalRank: O(n + E) since this function calls findImportantNodes, which is the most time complex task performed by findFinalRank.

- heapifyUp and heapifyDown: O(log n) since in the worst case, the program traverses the entire height of the tree which is at worst *log n* where *n* is the number of nodes in the tree.
- insert and extract: O(log n) as removing the root is O(1) and then performing resize functions such as heapifyUp and heapifyDown are also O(log n) time complexity.
- requestNCities: O(n log n + m log m) where *n* is the number of requested cities and *m* is the number of cities being inserted since the insert and extract functions have time complexity of *log n* and *log m*, respectively.
- addToMap: O(n) using id_city.find() which uses a linear traversal through all data points.
- addEdge: O(1) using adjList instant access and push_back, which are both O(1).
- outDegree(): O(E) since all the edges from adjList are checked once during the for loop.

**Overall Experience**

The project was a valuable learning experience that enhanced our understanding of complex algorithms and data analysis. We encountered challenges related to data volume and computational efficiency but overcame these through iterative refinement and collaboration.

**Challenges**

1. **Computational Overhead**

Calculating betweenness centrality for large networks was time-consuming and computationally expensive.

**Solution:** To manage this, we used heuristic approximations by focusing on high-congestion or high-degree nodes, which reduced the data size. We employed Brandes' Algorithm for its efficiency with large graphs.

2. **Data Integration**

Integrating and cleaning data from various sources was complex due to different formats, units, and missing values.

**Solution:** We developed a robust data preprocessing scheme using pandas, which included standardizing formats, addressing missing values, and ensuring data consistency.

3. **Adapting Real-World Data:**

Adapting diverse real-world traffic data for meaningful analysis was difficult due to the various attributes and precision of coordinates. Traffic data includes multiple attributes such as start location coordinates, congestion severity, duration, and distance, which need to be effectively utilized and adjusted for graph modeling.

**Solution**: We focused on relevant attributes for our analysis and simplified the model by assuming traffic jams can stretch in cardinal directions to affect neighboring cities if the distance is long enough for the obtained coordinates to map to a new city. We had to sacrifice the coordinates' precision to one decimal point due to limited freely available mapping data. However, this level of precision, translating to approximately 6 miles, still provided a satisfactory result for mapping and analysis purposes.

4. **Weight Calculation and Visualization:**

Calculating accurate edge weights and visualizing congestion ranks on a map required careful representation and normalization.

**Solution:** We derived congestion ranks by summing the severity weights of each congestion event and storing them in a map. The final rank combined congestion and centrality scores using a heuristic approach, focusing on key nodes based on degree and congestion. For visualization, we applied z-normalization to standardize congestion levels across cities, setting the median to 0. Heatmaps and color-coding were used to clearly represent these normalized ranks on the map.

**Future Changes:**

- Incorporate Additional Factors: Plan to integrate factors such as population density, growth predictions, and construction costs, with research to determine appropriate weighting.
- Budget Distribution: Add functionality to allocate budgets based on city prioritization for more effective resource management.
- Detailed Street-Level Data: Future updates will include street-level data for more granular analysis and visualization of congestion and infrastructure needs.

**If we were to start again, we would:**

- Early Implementation of Heuristic Methods: Begin with heuristic approximations to manage computational load more effectively.
- Enhanced Data Preprocessing: Streamline data merging and cleaning processes to reduce preprocessing time.

**Individual Learnings:**

- William Johnson: Gained deeper insights into C++ and data structure design.
- Anhelina Liashynska: Developed expertise in data preprocessing in Pandas, Gained deeper insights into C++ and data structure design.
- Daniel Lopez: Improved skills in algorithm implementation and interface design.