MULTIGRAPH: IMPLEMENTATION

2nd Project, Part 2

Group E
Henrique Sousa
Mateus Silva
Melissa Silva

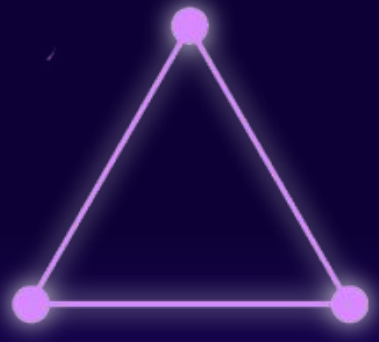
01

RECAP

Here’s a refresher of our last presentation.

Data Structure: Multigraph

- Graph that allows multiple edges between any two vertices;
- To put it simply: multigraphs allow for **multiple connections between the same pair of vertices**;
- Each edge in a multigraph is associated with a pair of vertices, and may include a weight or cost associated.



Simple Graph

Multigraph

DayofMonth	DayOfWeek	Carrier	OriginAirportID	DestAirportID	DepDelay	ArrDelay
19	5	DL	11433	13303	-3	1
19	5	DL	14869	12478	0	-8
19	5	DL	14057	14869	-4	-15
19	5	DL	15016	11433	28	24
19	5	DL	11193	12892	-6	-11

Flights dataset

airport_id	city	state	name
10165	Adak Island	AK	Adak
10299	Anchorage	AK	Anchorage International Airport
10304	Aniak	AK	Aniak Airport
10754	Barrow	AK	Wiley Post/Will Rogers Memorial
10551	Bethel	AK	Bethel Airport

Airport dataset

lat	lng
42.2327	-83.3412
40.7862	-111.9820
45.5867	-122.5870
38.7414	-90.3647
39.0571	-84.6625

Extra data

distance	flight_time
1844.869	2.291
3201.109	3.976
1012.473	1.257
710.184	0.882
3056.374	3.796

Flight distance and duration

Dataset

- We have 2 files: flights.csv and airports.csv.
  - Flights has 2.7M rows and 7 columns;
  - Airports has 365 rows and 4 columns.
- We looked for extra data to find out the duration and distance of each flight.

There is more to be said about our dataset - but that's for later...

Problem Definition

1

Path-Finding

Find the shortest path: a list of edges connecting airports A and B.

2

Spanning Tree

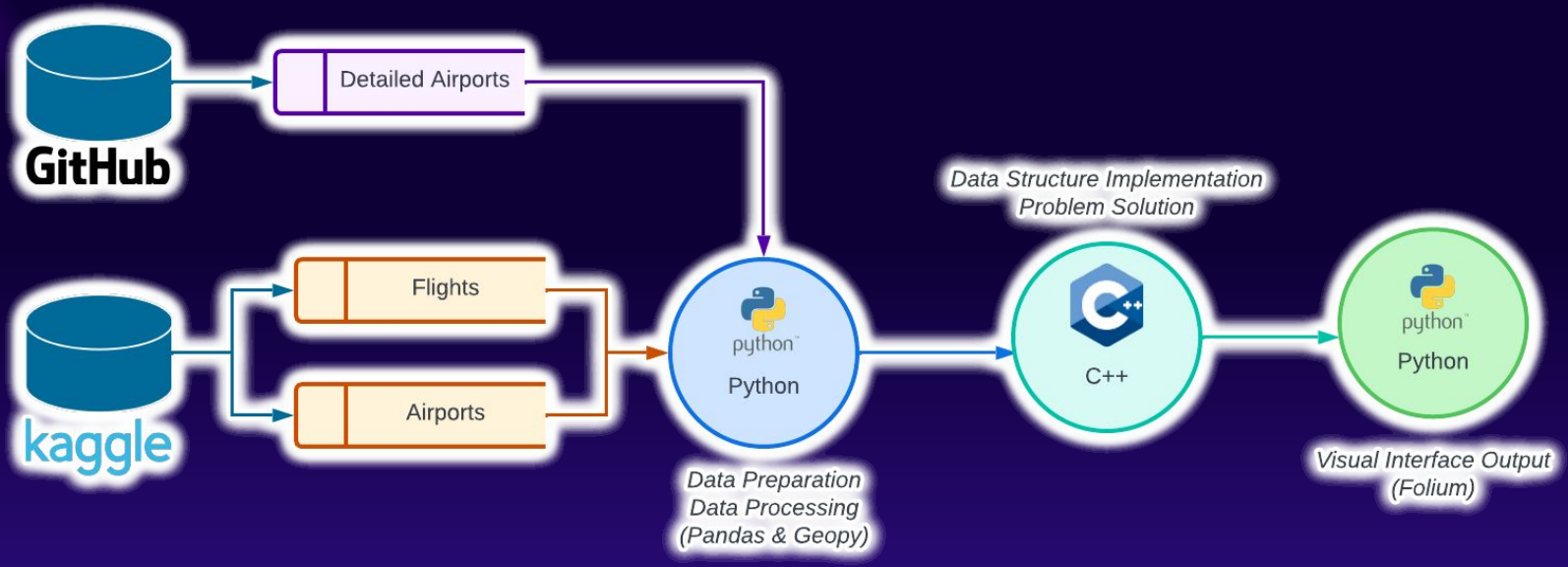
Find a spanning tree comprising the best edges to connect all nodes in the data set.

3

Erdős Number

Find the least amount of edges required that connects two airports with the least amount of connection flights.

Here's how we went about it...



Pipeline diagram.

02

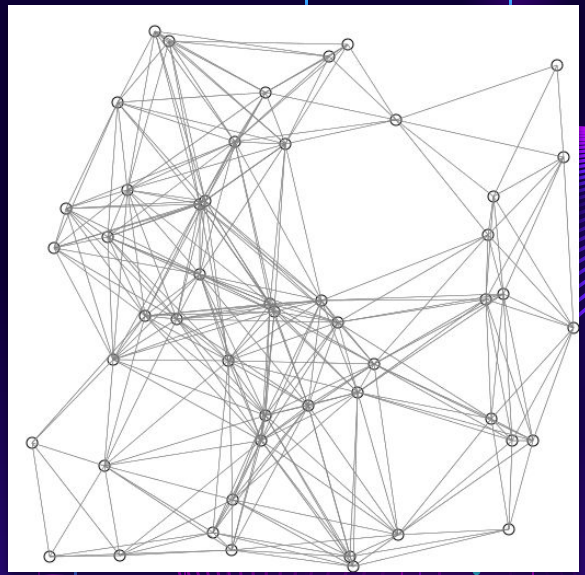
Implementation

Details on how we implemented our idea.

Dijkstra

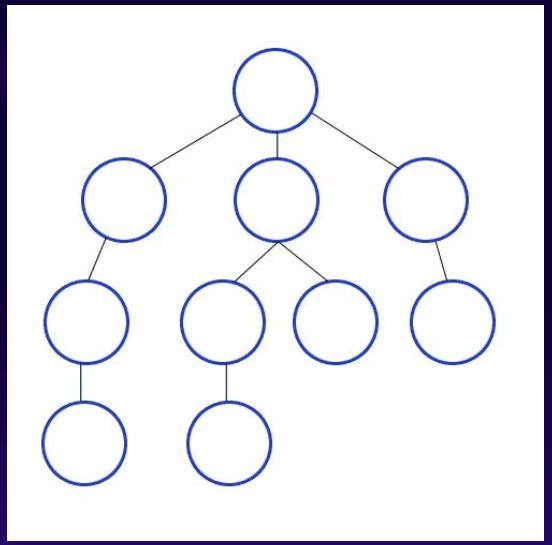
- Dijkstra is capable of solving our path-finding problem;
  - From a given node, it'll explore its edges, finding all reachable nodes starting with the adjacent ones and going further;
  - It uses a priority queue, where minimum distance is considered;

- To improve efficiency, we made a variation of the algorithm: ***Dijkstra By Node***;
  - It considers the multigraph's core characteristic: the possibility of multiple edges between a pair of nodes;
    - This mean one node could be pushed to the queue as many times as there are edges leading to it;
  - But, since distance between nodes (airports) was used, the above thing never happens.



Visual execution of the algorithm.

Breadth-First Search (BFS)



Visual execution of the algorithm.

- BFS is a recursive algorithm for finding all the vertices of a tree or graph data structure;
  - The algorithm marks visited nodes as it finds them;
  - It can also be used to compute the shortest distance to any reachable vertex; *
- This algorithm was used to solve the Erdős Number problem;
  - The algorithm calculates the shortest path between the origin and destination nodes, returning a list of edges - the answer to the problem is the length of this list;
- It was also used in the Minimum Spanning Tree problem, for testing if a graph is connected;
- We also created a **BFS by Node**;
  - This variation iterates over the subsets of edges that connect to each destination node, one by one.

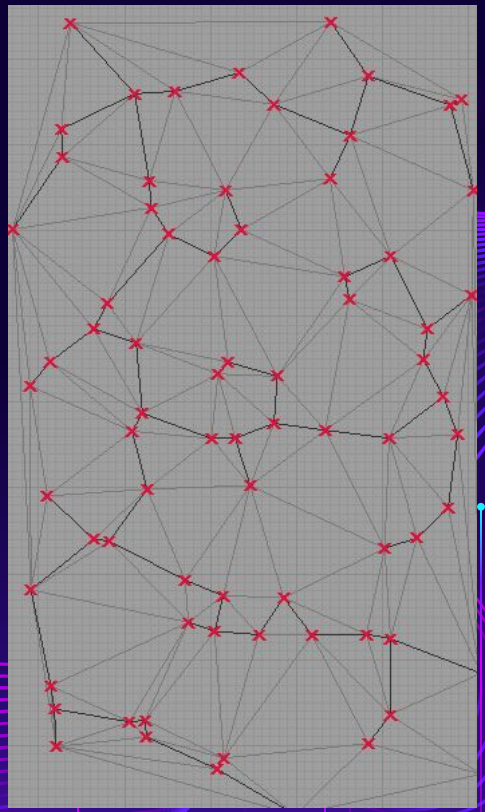
** when applied to unweighted graphs;*

Reverse Delete – Minimum Spanning Tree

- Reverse Delete Algorithm is used to assemble a Minimum Spanning Tree (MST) in directed graphs and it's also **recursive**, like BFS;
  - Edges are ordered by decreasing weight and removes the first one;
  - Then, the graph is tested for connectivity:
    - If it is, then the edge *isn't* part of the MST;
    - Otherwise, the edge *is* part of the MST;

- Our connectivity check goes beyond weak connectivity;
  - In a directed graph, connectivity must guarantee that all other nodes are reachable from the root;
  - If an edge is removed, losing connectivity implies that there are unreachable nodes;

- The spanning tree of a graph of *N* vertices has *N - 1* edges. A complete directed graph of *N* vertices has *N* ✕ *(N - 1)* edges;
  - To improve performance, a virtual graph is made, composed of the edges with the smallest weights;



A changing spanning tree.

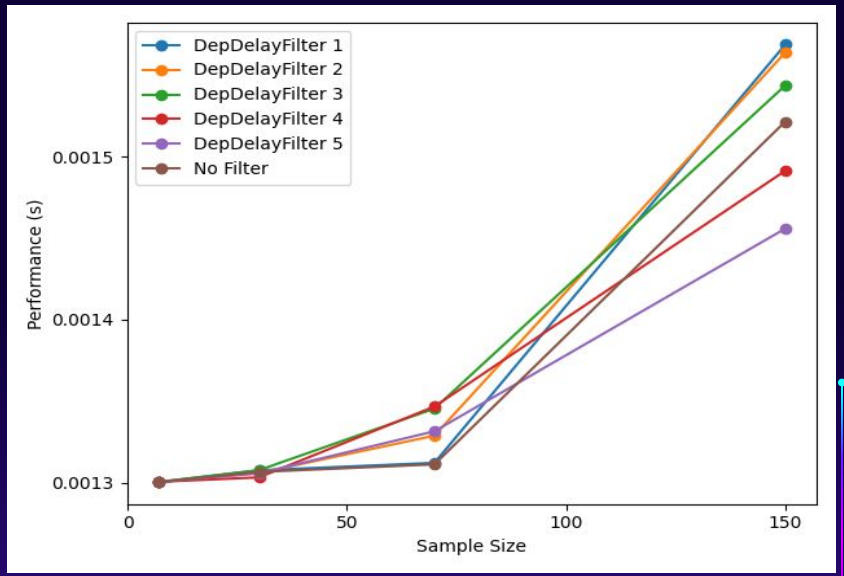

03
Stress Study
Testing the limits
of our work.

Different Data Samples

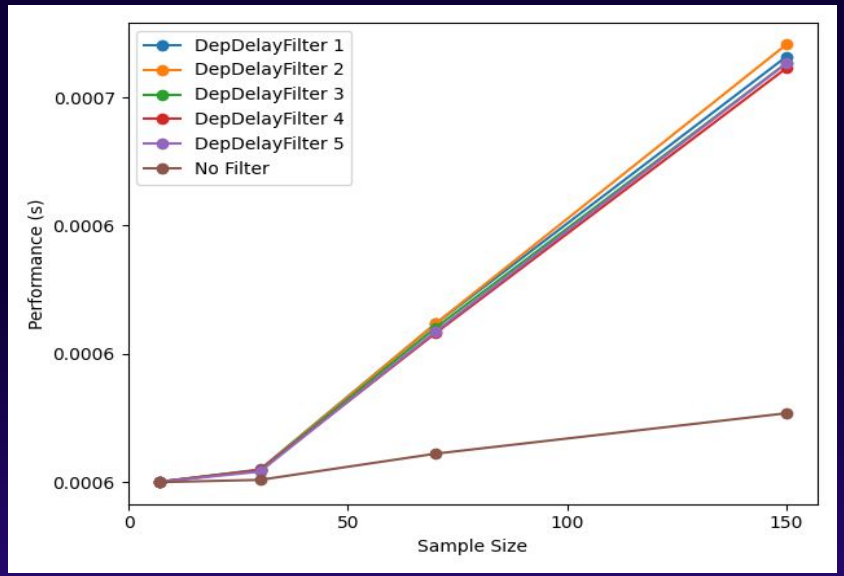
- We scaled the data set to test how our implementation would fare in the real world;
  - There are 5000 public airports in the USA;
  - The data set only has 365, out of which only 70 airports include flights;
  - We got data from 704 USA airports;
    - Airport IDs were generated randomly but each has a unique one;

- We test sample sizes of **7**, **30**, **70** and **150** airports;
  - For each size, we generated a flights file randomly - values from the original file were used a base, and we settled on line counts using a mathematical expression related to the original file's **connection density**: *1118N x [(N-1)/2]*

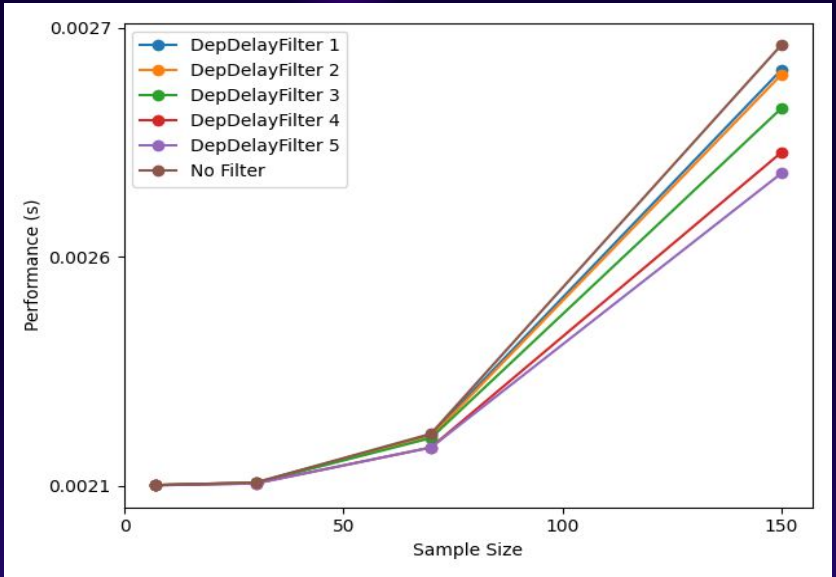
Performance



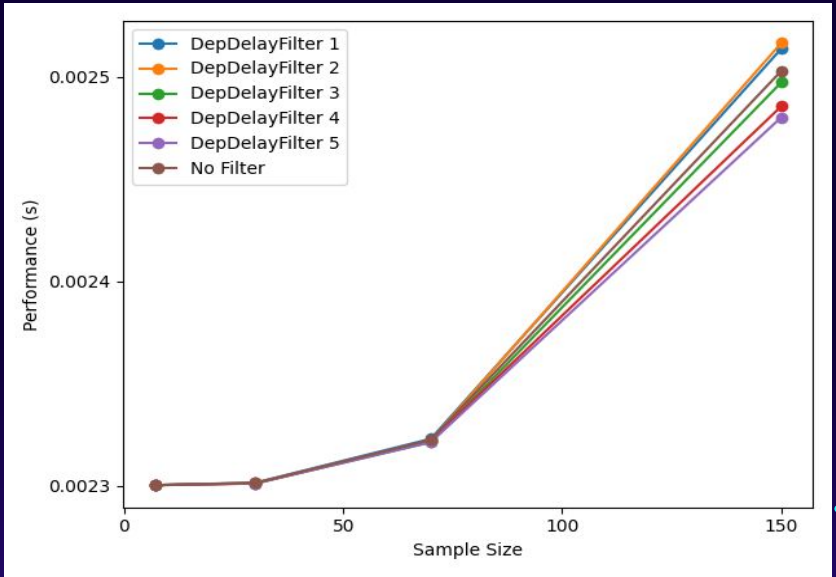
BFS



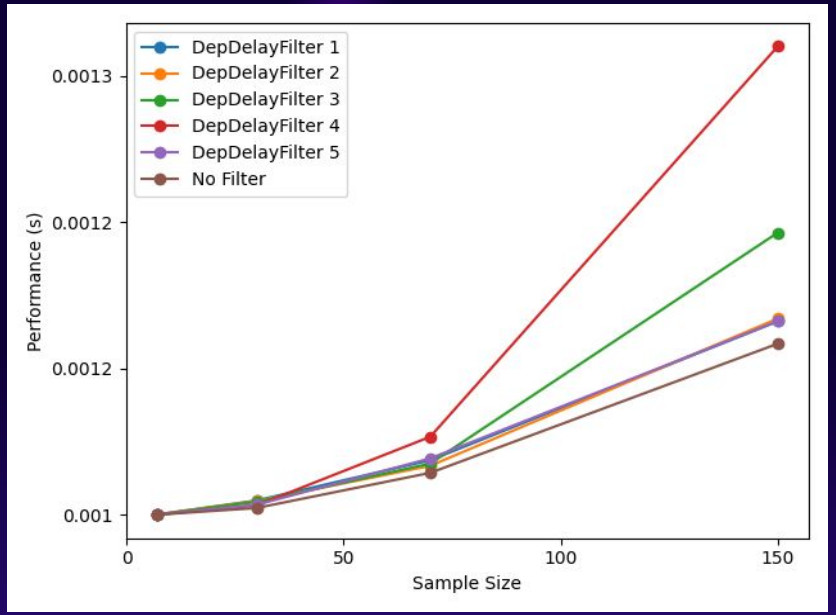
BFS By Node



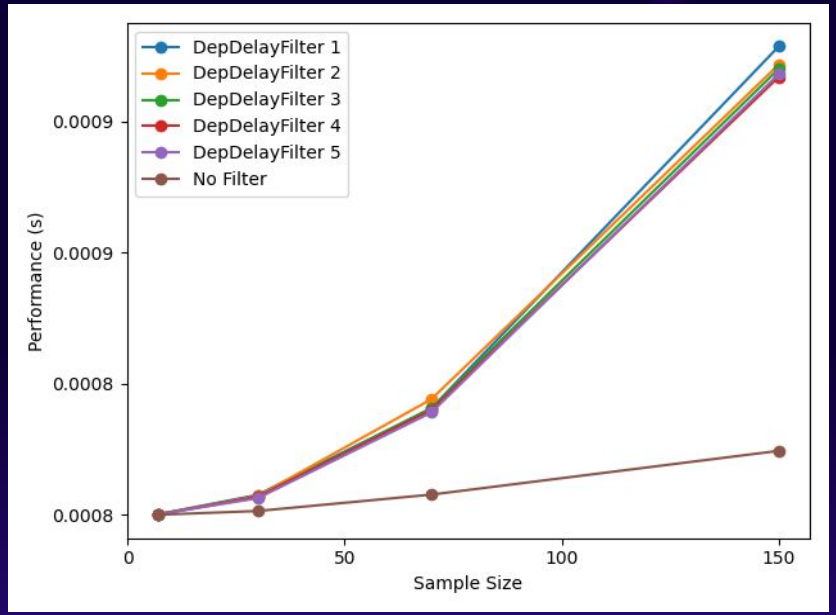
Dijkstra



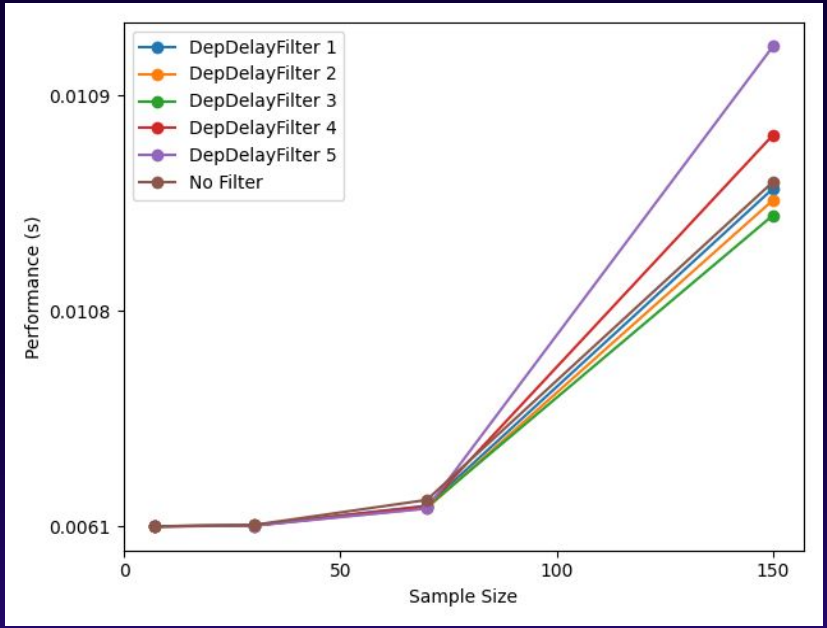
Dijkstra By Node



Erdős



Erdős By Node



Spanning Tree

Overall...

- For every algorithm, execution time grows proportionally to the sample size: algorithms are at their fastest with our 7 airports sample and at their slowest with 150 airports sample;
- Alternative versions to the algorithm created were successful in improving performance;

04

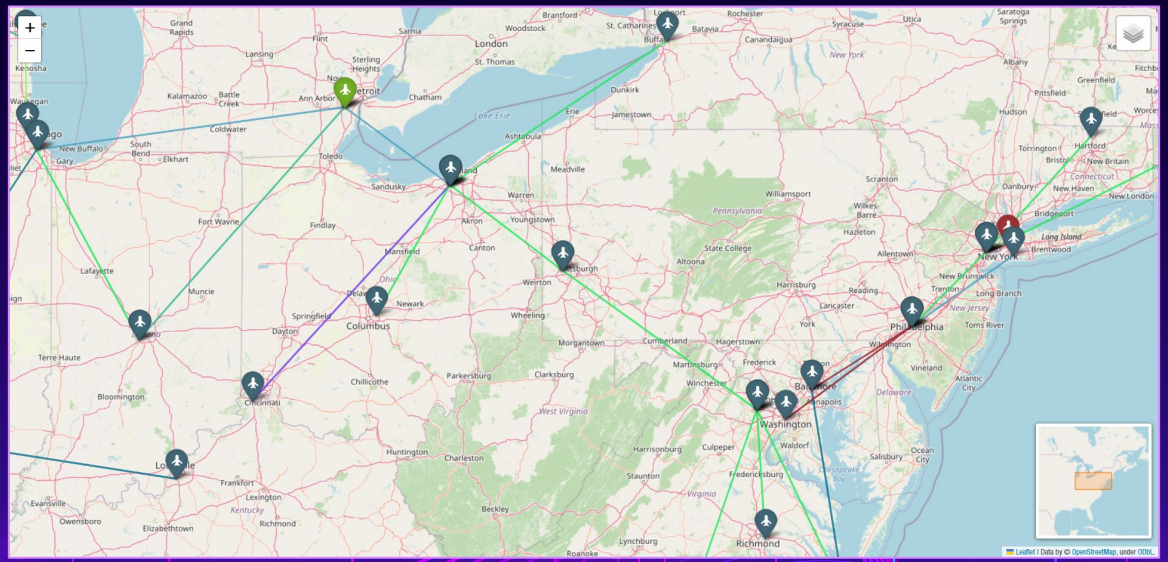
Demonstration

A short video demo
of our work.

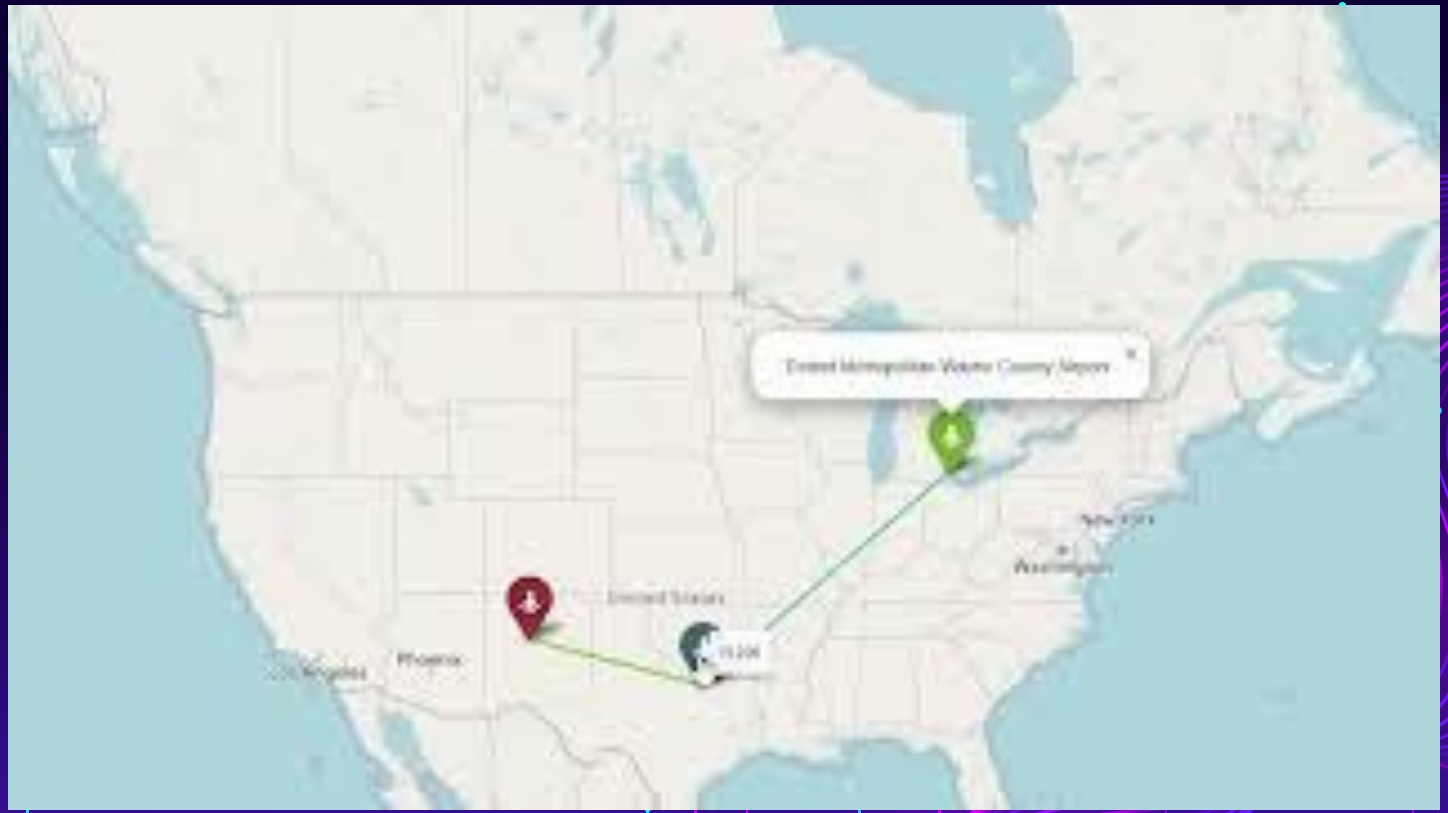
As proposed, we made a GUI, but to make it a little more transparent:

- **Green** is used for the origin node;
- **Red** is used for the destination node;
- **Blue** is used for intermediate nodes;
- **Edges** are colored by carrier;

Please note that the minimum spanning tree problem doesn't really have a "destination" or an "origin" - the first and last nodes iterated take those colors.



Demo



Conclusion

- We are satisfied with how the project turned out, as were able to follow through with our proposals with very minor changes;

- We’ve acquired new skills and believe we focused on the improvement of our implementation, which leaves us proud of what we achieved in the project.

Do you have any questions?