Algorithms Group Project

Group Y

Management and Artificial Intelligence
Prof. Irene Finocchi
Prof. Miro Confalone

Tommaso Moriondo (316331)
Alessandro Torre (309281)
Eleonora Raimondo (321521)
Bianca Marino (321971)

11 May, 2025

Task 1 part 1 - Temporal Analysis

Temporal Analysis

- **1.1.1** Analyze the number of monthly flights and represent it graphically.
- **1.1.2** Analyze the number of flights per day of the week, broken down by month (e.g., January: Monday X flights, Tuesday Y flights, etc.).
- **1.1.3** Analyze how flight routes vary by season, grouping months into Winter (Dec - Feb), Spring (Mar - May), Summer (Jun - Aug), and Autumn (Sep - Nov).
- **1.1.4** Analyze whether airlines influence traffic on the most frequently used routes (e.g., which airlines operate the busiest routes?).

Route Analysis

- **1.2.1** Identify the busiest flight routes based on the "origin-destination" pair and represent them graphically.
- **1.2.2** Calculate the average flight duration for each route and compare it with the distance to classify them into short, medium, and long-haul flights.

Delay Analysis

- **1.3.1** Analyze the distribution of departure and arrival delays to understand their frequency and severity.
- **1.3.2** Identify the time slots with the most delays for each day of the week and month to detect possible patterns.
- **1.3.3** Analyze delays based on route and airline, comparing the performance of different companies.

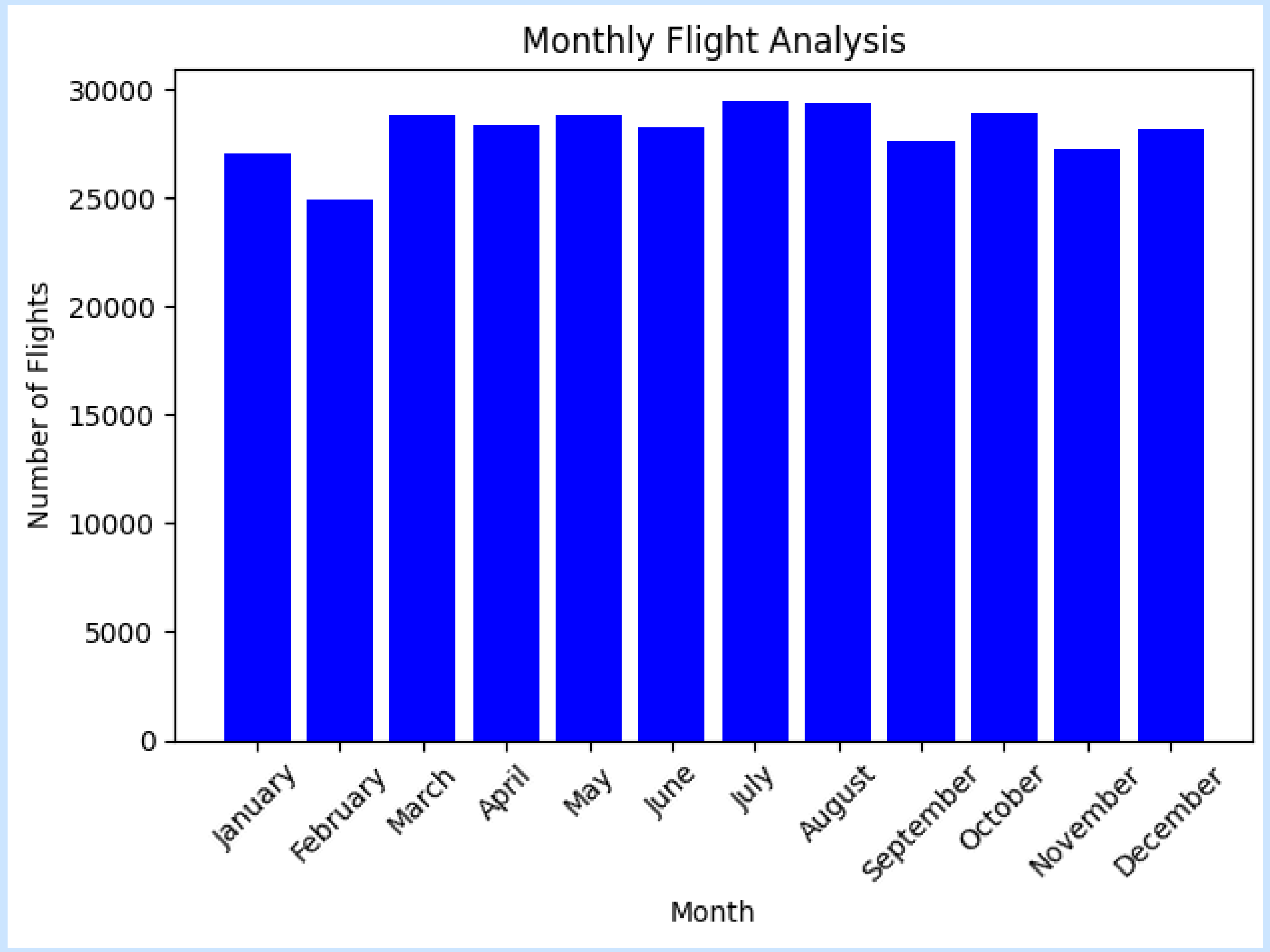
Task 1 part 2 - Algorithm analysis

We will observe 4 alghoritms:

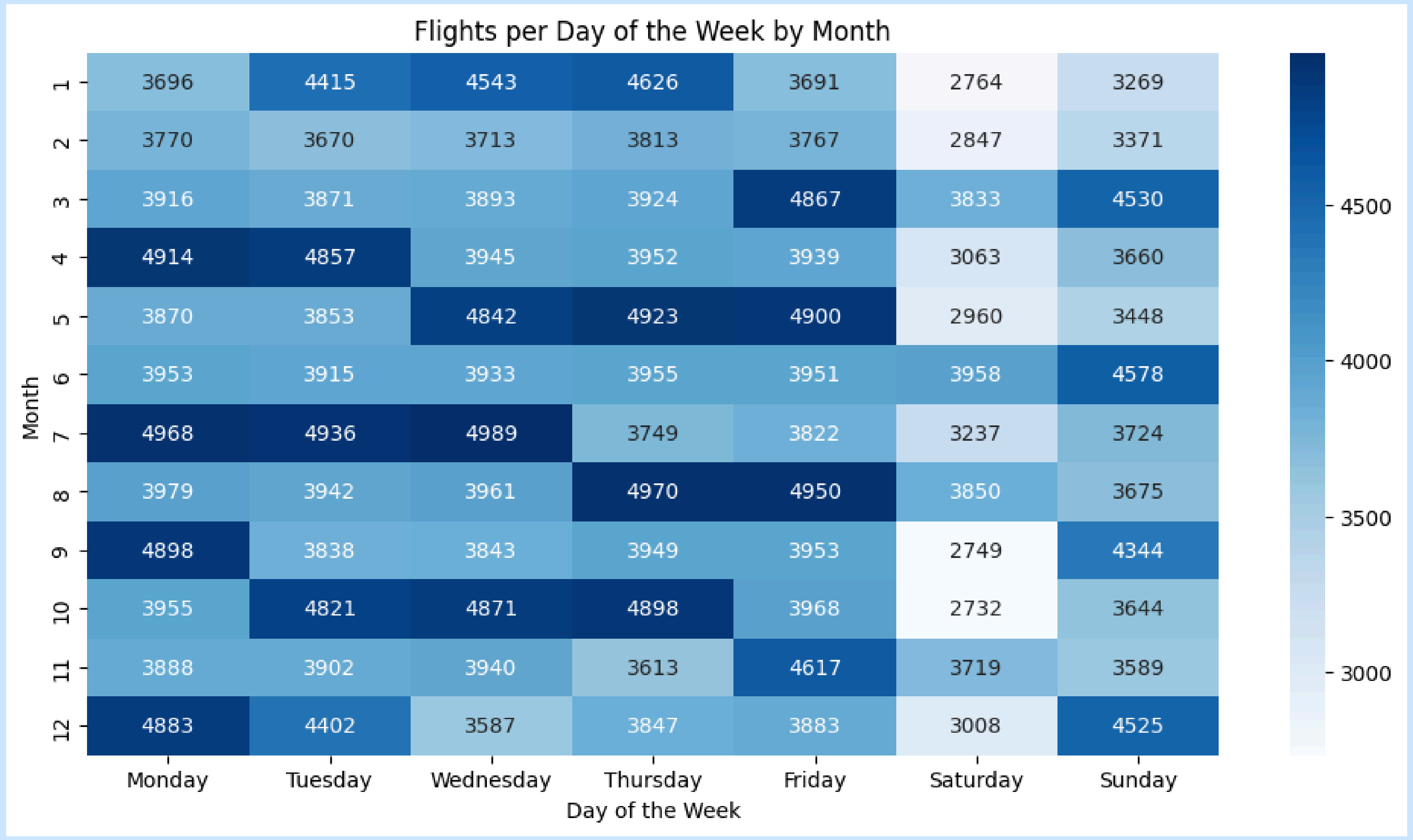
- **Bubble sort:** Time Complexity ($O(n^2)$)
- **Insertion sort:** Time Complexity ($O(n^2)$)
- **Merge sort:** Time Complexity $O(n\log(n))$
- **Quick sort:** Time Complexity $O(n\log(n))$

Task 1 - Part 1 - Temporal Analysis

Task(1.1.1)



Task(1.1.2)



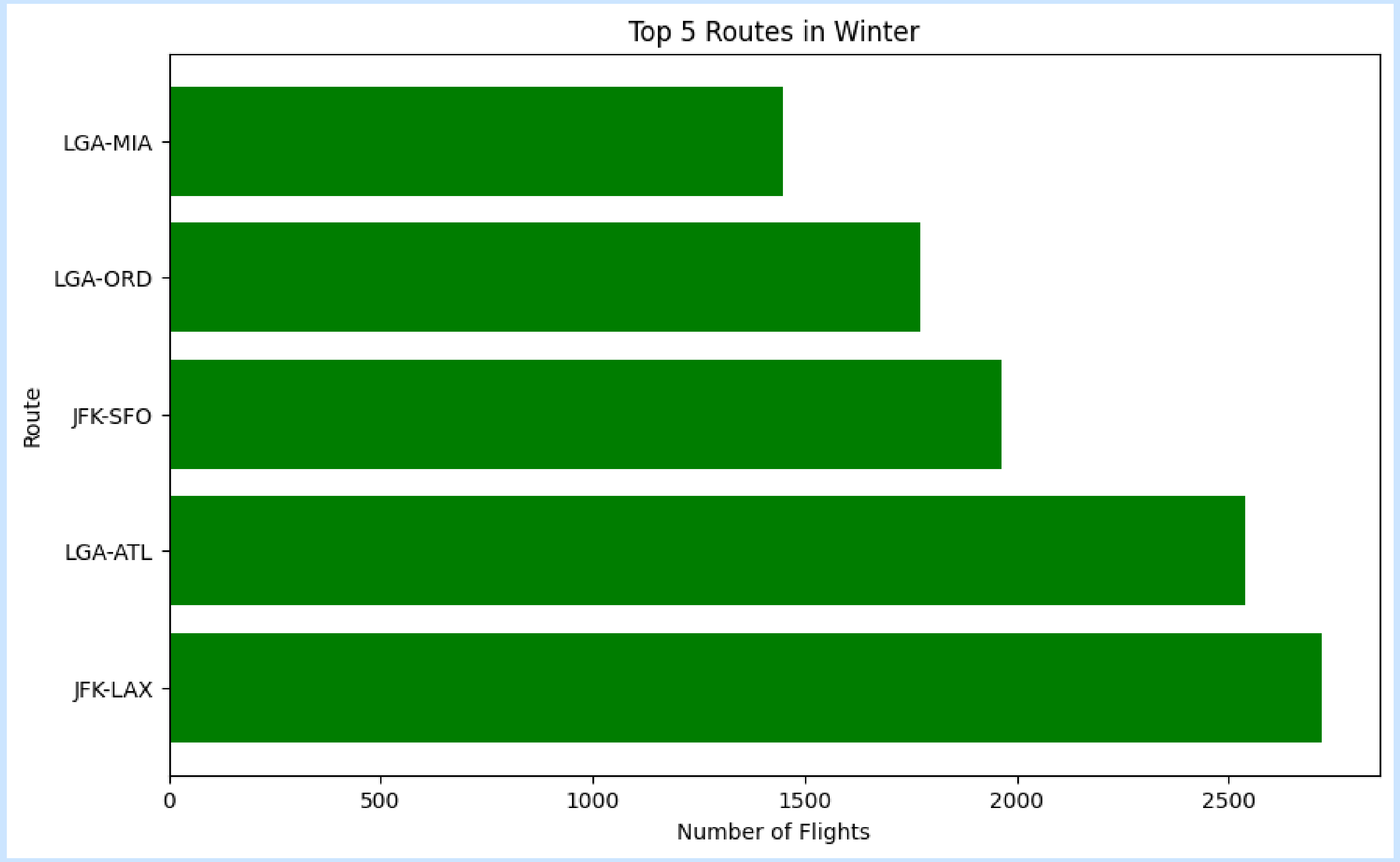
Seasonality and Monthly Trends

- Flight peaks: highest in July (29,425) vs lowest in February (24,951)
- typical travel behaviors due to weather conditions/work breaks

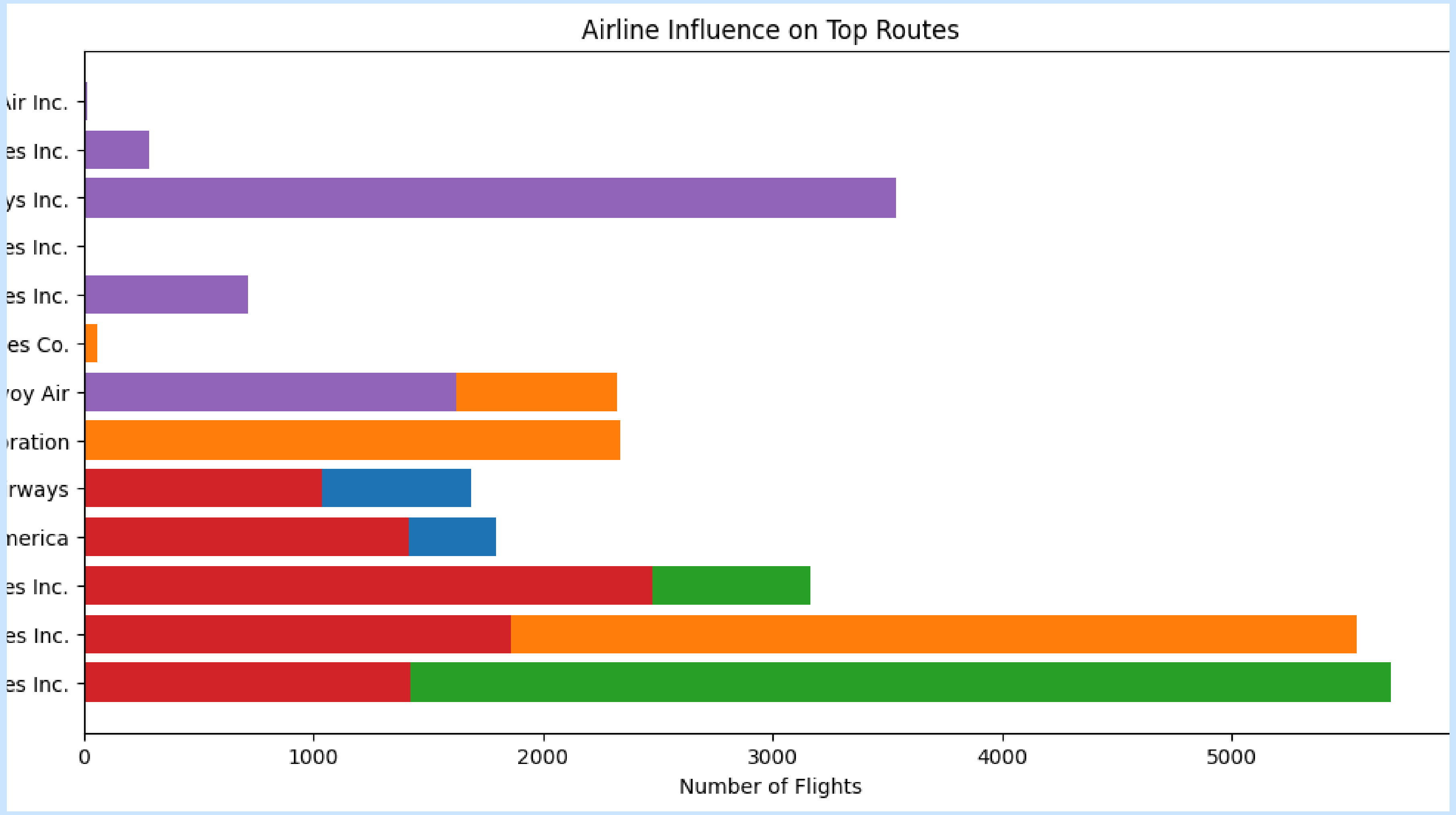
Weekly Distribution

- Flight frequency: highest on & Thursday of July & October (record with Wednesday in July)
- Flight frequency: lowest Saturdays consistently - perhaps preferred arrival before the weekend
- high frequency for Mondays in April, July, September, December (business travel/weekend returns)

Task(1.1.3)



Task(1.1.4)



Route Trends by Season

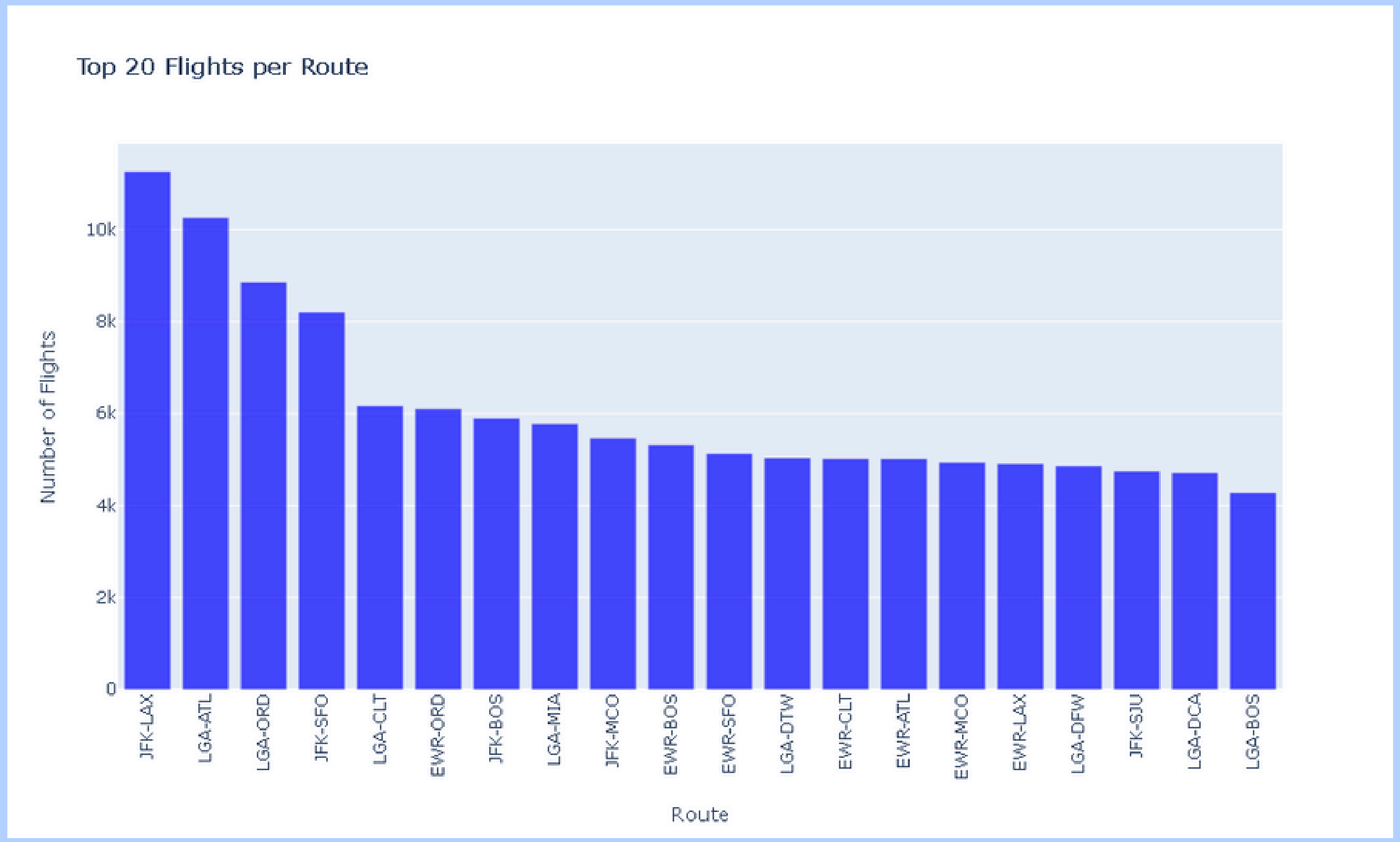
- dominating air traffic: JFK-LAX
- low-ranked routes: EWR-ORD
- Observations: high population density, economic importance of these cities

Airline Influence on Busy Routes

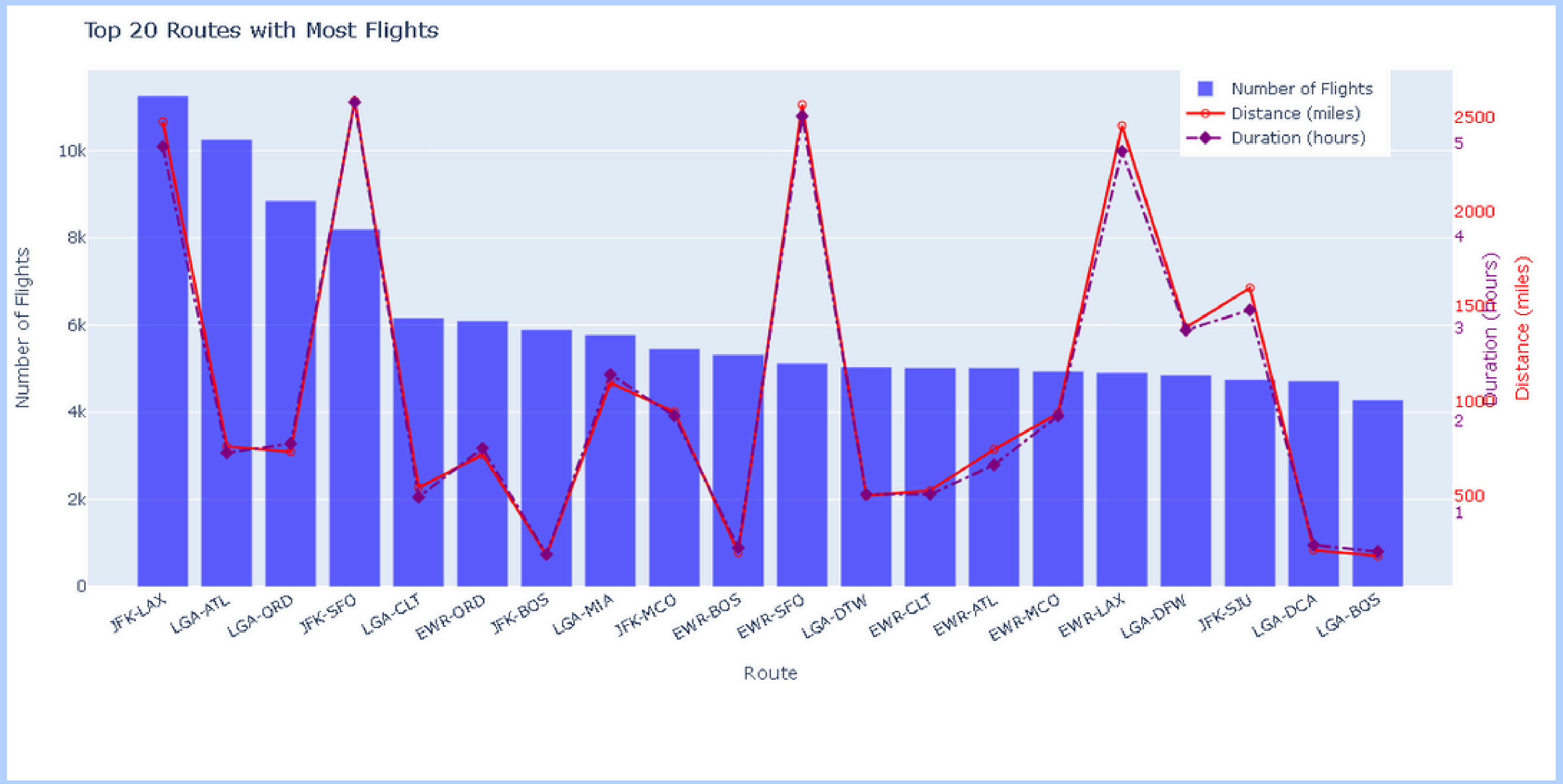
- Delta Air Lines: LGA-ATL 54% of flights
- American Airlines: LGA-ORD 64% of flights
- US Airways: LGA-CLT 57% of flights
- Observations: near-monopolies on specific routes through frequency advantages

Task 1.2 - Temporal Analysis

Task(1.2.1)

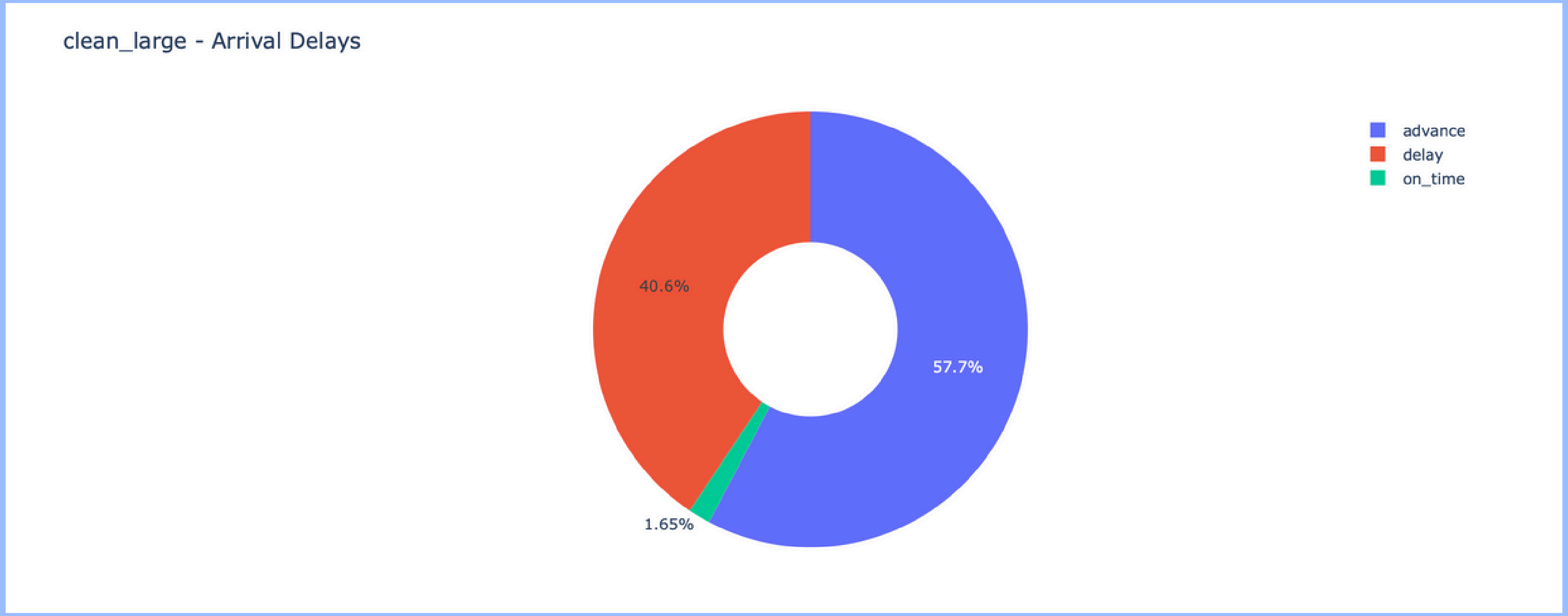


Task(1.2.2)

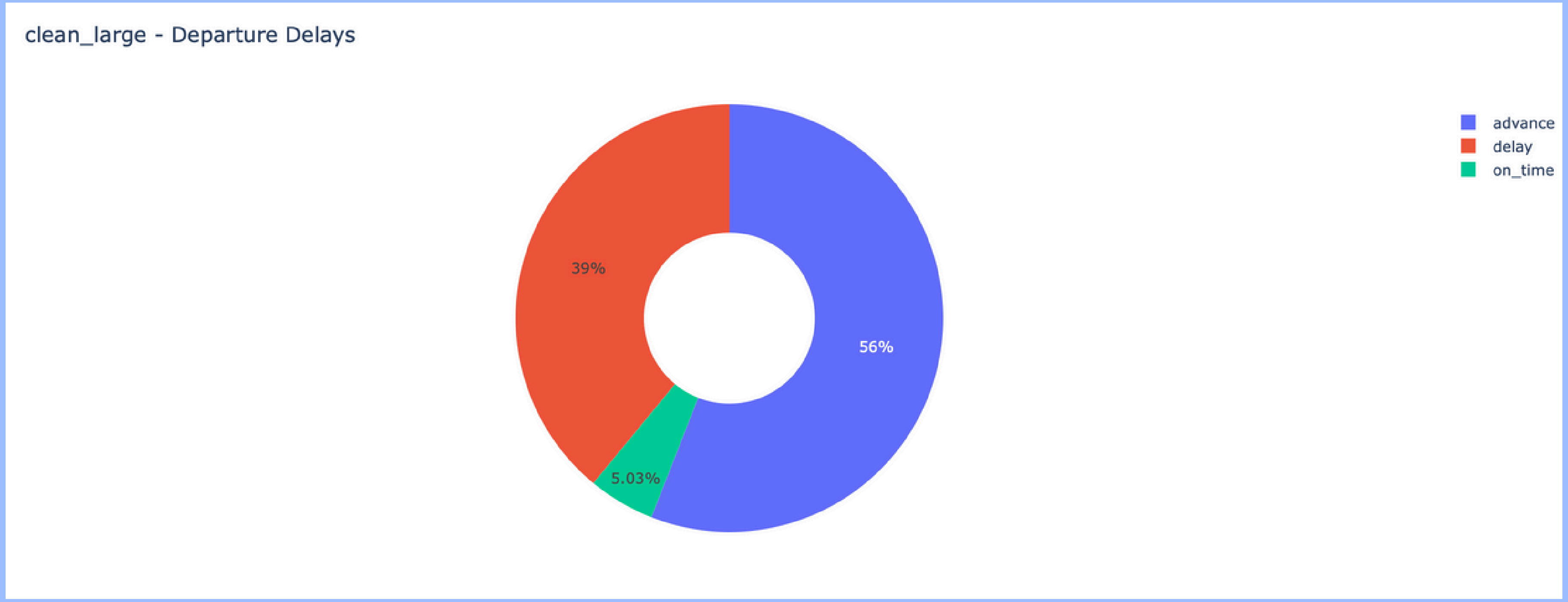


- JFK-LAX, LGA-ATL, and LGA-ORD are the top 3 busiest routes, each exceeding 8,000 to 11,000 flights.
- Major NYC airports (JFK, LGA, EWR) dominate the busiest routes list, highlighting their key role in national air traffic.
- The most frequent routes connect major business and hub cities, emphasizing heavy commuter and business travel patterns.

Task 1.3: Delay Analysis – Comprehensive Conclusion



Metric	Departure	Arrival
Mean Delay (minutes)	39.0	41.0
Median Delay (minutes)	19.0	21.0



Key Observations

Skewed distributions occur when average delays exceed the median, often due to a few extreme delays while most are mild.

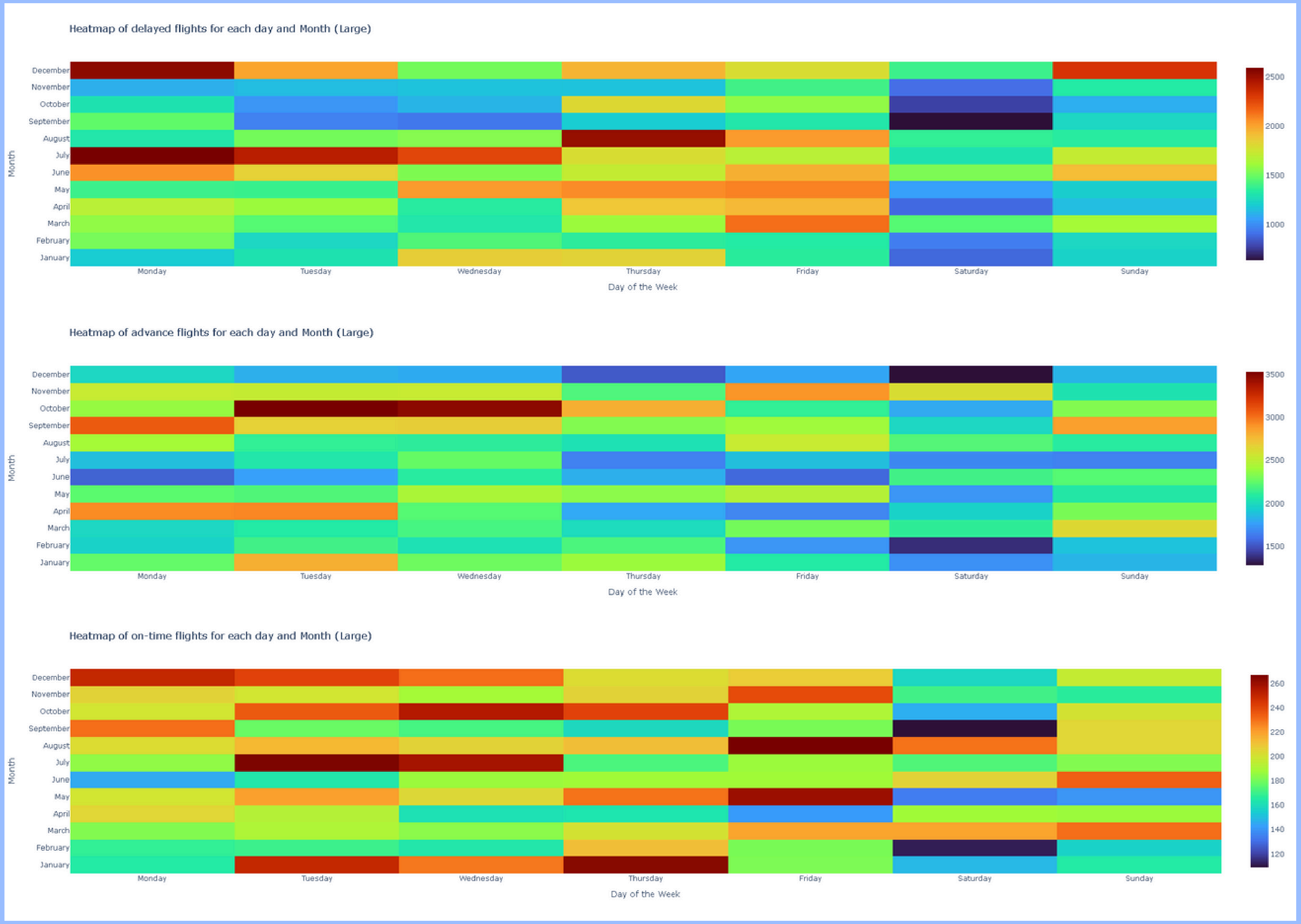
Less than 5% of departure flights and almost none of arrival flights are on time.

Over 55% of both departures and arrivals take place early, suggesting schedules allow extra time for potential delays. **Long tails in delay distributions** show that a small number of flights with delays over two hours significantly impact overall performance metrics.

Conclusion

Most flights have early or moderate delays, but average delays can be misleading without considering their distribution. Airlines may need to reassess schedule **padding vs actual performance.**

1.3.2 Delay Patterns by Time of Day, Week, and Month



Flight performance varies across time.

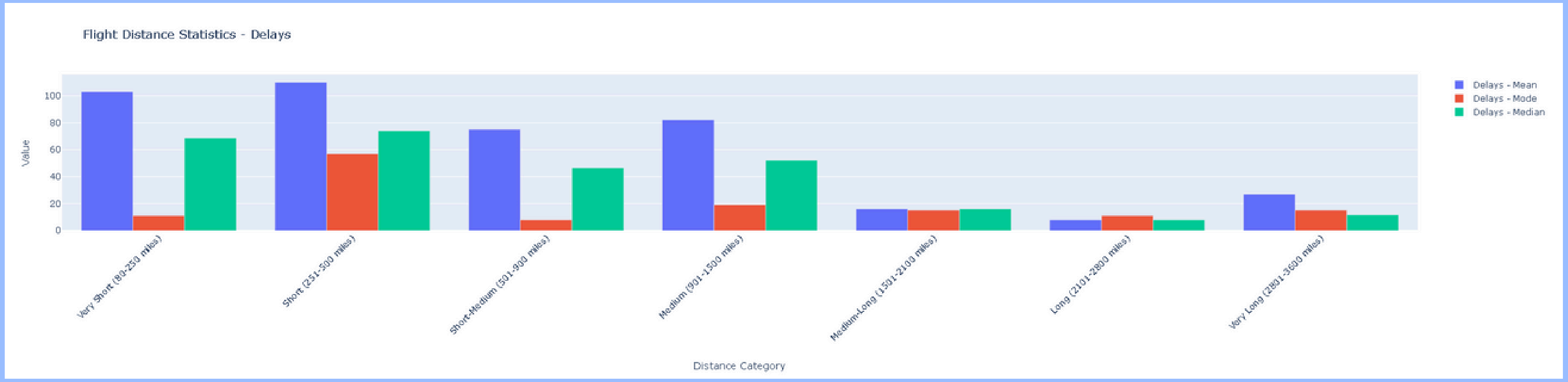
Delays peak in July and December, especially on Mondays and Thursdays, likely due to holiday and peak travel periods.

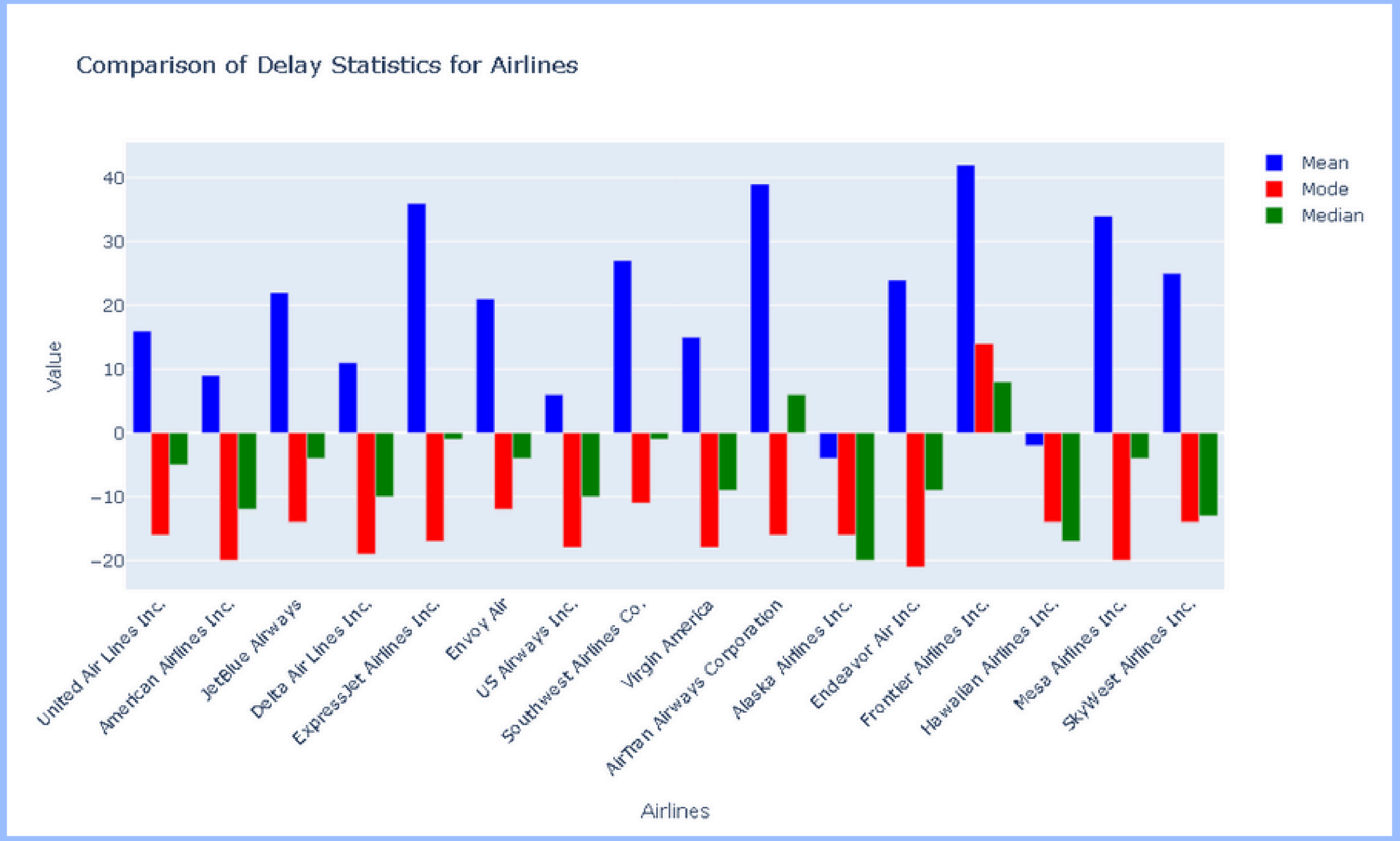
Early arrivals are most frequent, especially on Saturdays and in October and early-year months, suggesting lower traffic congestion.

On-time flights remain rare and uneven, confirming that traditional punctuality metrics underrepresent actual airline performance."

Most flights do not arrive on time, often being early or delayed, especially during holidays. This shows the need for a nuanced understanding of airline punctuality, considering both extreme delays and early arrivals in performance evaluations.

1.3.3 Delay Analysis by Distance and Airline





Key Findings:

- **Airline Performance:** Average flight delays vary significantly among airlines.
- **Short-Haul Issues:** Flights under 900 miles tend to have higher average delays due to quick turnarounds and airspace congestion.
- **Inconsistent Short Flights:** Delay patterns for shorter flights are less predictable.
- **Long-Haul Reliability:** Longer flights usually have lower delays and better punctuality, likely due to scheduling buffers.

Key Takeaway:

Flight distance is a major predictor of delays: shorter flights face more frequent and variable delays.

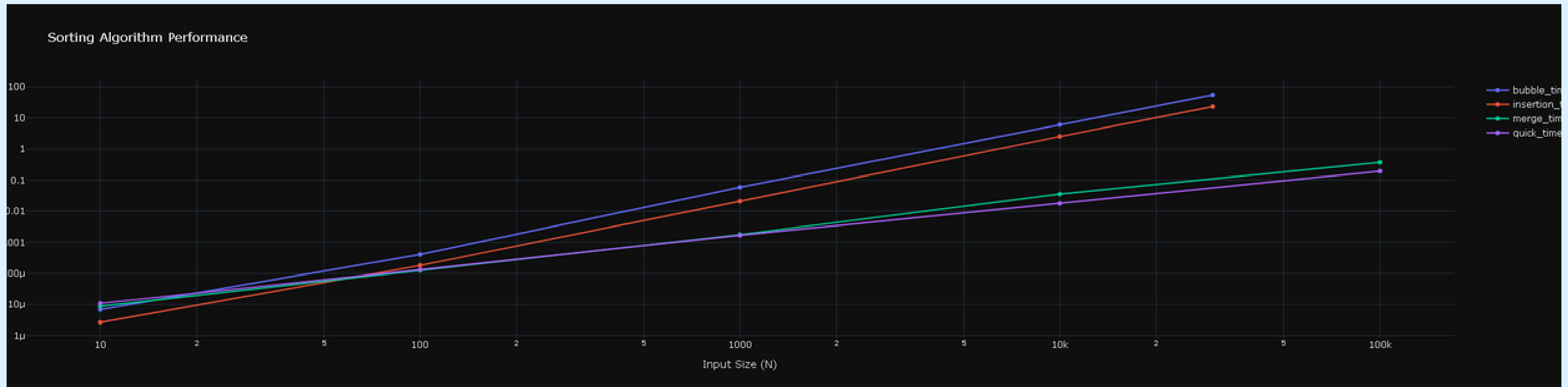
Airline-specific factors also contribute significantly to delay performance.

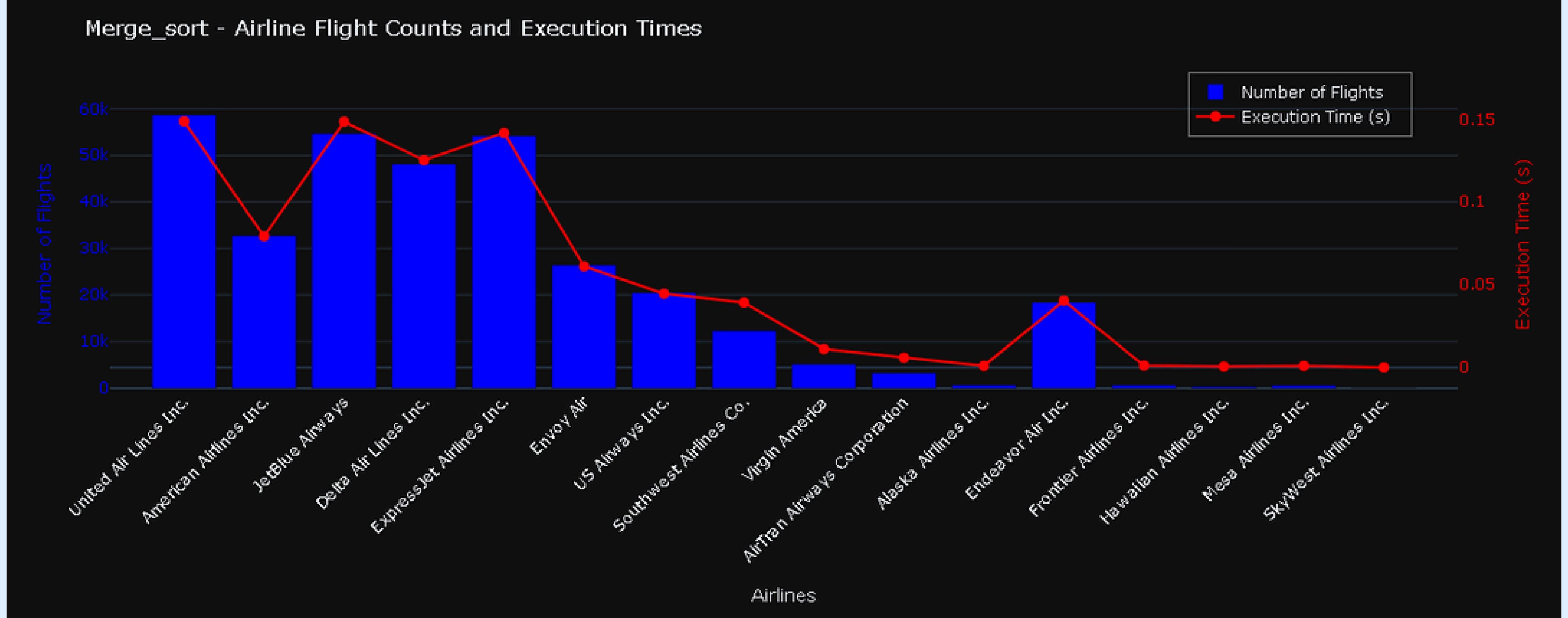
Key Findings & Insights:

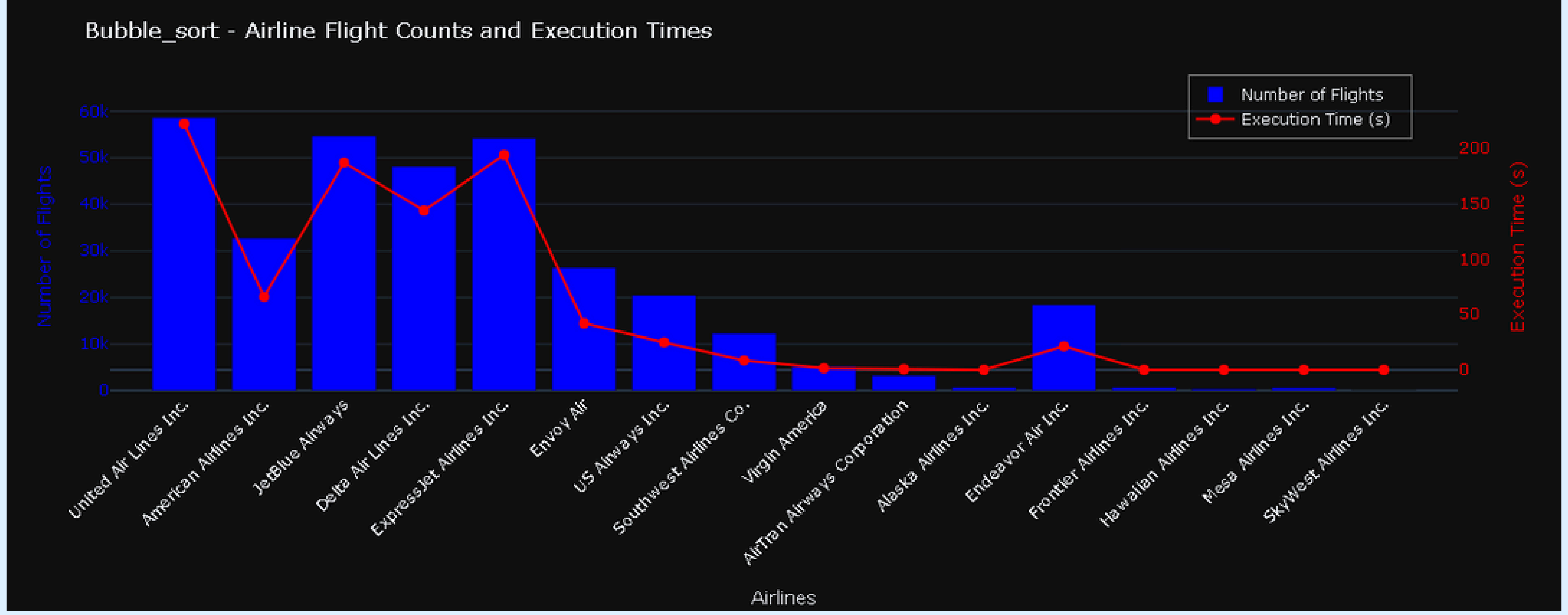
- **Seasonal & Weekly Trends:** Significant variations in delays and on-time performance based on the time of year and day of the week.
- **Short-Haul Vulnerability:** Flights under 900 miles are particularly susceptible to higher and more variable delays.
- **Long-Haul Reliability:** Longer flights tend to be more punctual due to potential scheduling buffers.
- **Airline Impact:** Operational efficiency differs across airlines, contributing to varying delay statistics.

Project Outcome & Recommendations:

- Developed data-backed recommendations for **airlines**, **airports**, and **regulators**.
- Focused on optimizing **short-haul operations**, adapting scheduling, and improving passenger communication.
- Emphasized the need to address factors beyond just volume to reduce delays (e.g., airport efficiency).







Task 1 - Part 2- Algorithm analysis

Core Findings:

- **O(n²)** Algorithms (Bubble & Insertion): Execution time grows rapidly with input size (see steep curves). Inefficient for large datasets.
- **O(n log n)** Algorithms (Merge & Quick): Scale much better for larger inputs (flatter curves). More efficient.

Asymptotic Notation Matters:

- **O(n²)**: Algorithms like Bubble Sort and Insertion Sort have a quadratic time complexity. Their execution time increases proportionally to the square of the input size (n). This makes them **impractical** for large datasets.

- **O(n log n):** Algorithms like Merge Sort and Quick Sort exhibit a logarithmic-linear time complexity. Their execution time **grows more slowly** with increasing input size, making them significantly **more efficient** for larger datasets.

Task 2 – Search & Data Structures Performance

Linear vs Binary Search

- **2.1.1** Compare the performance of:
  Linear Search (O(n))
  Binary Search (O(log n))
  On sorted flight data.

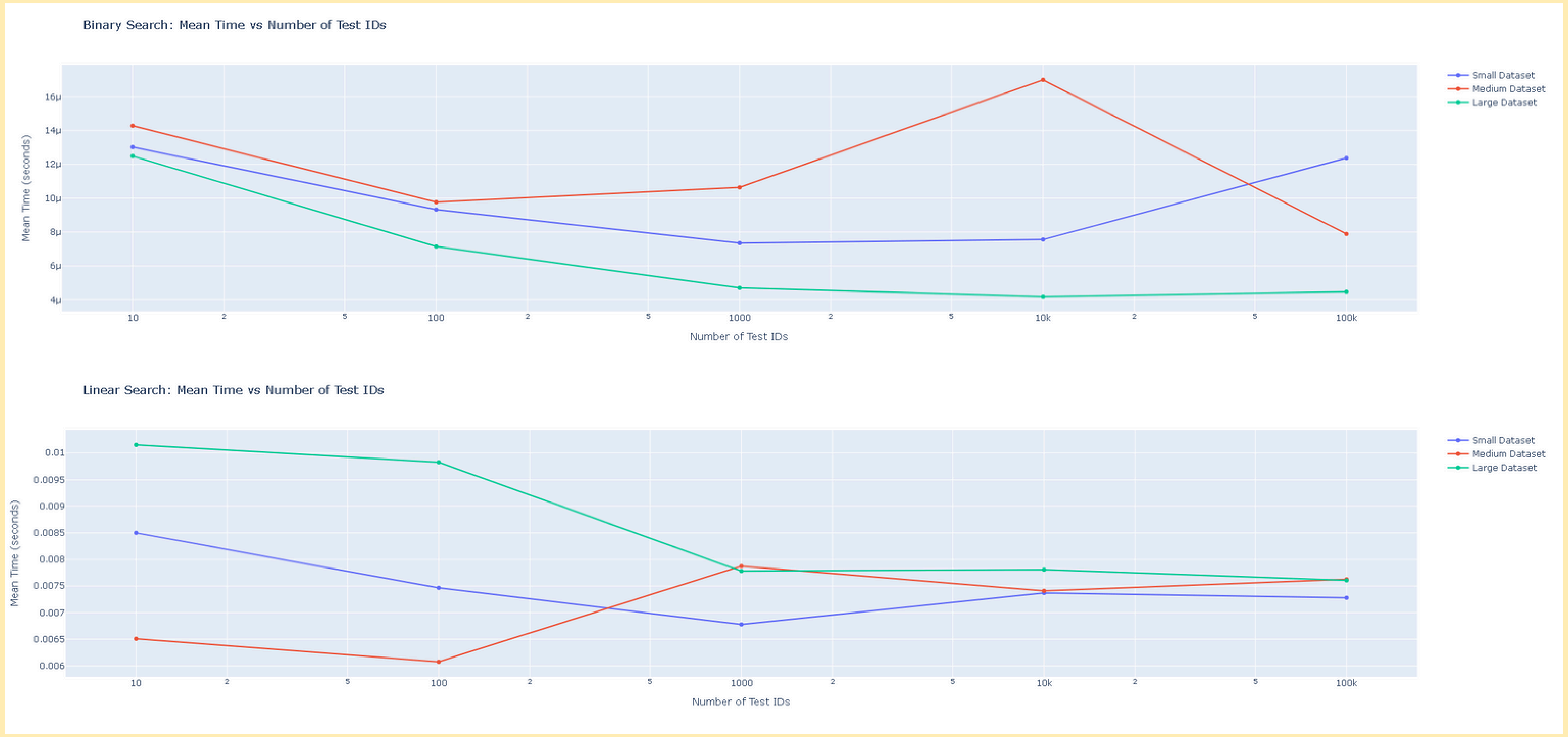
Hash Table Lookup

- **2.2.1** Store flight records in a hash table (dictionary), using flight IDs as keys.
- **2.2.2** Enable constant - time O(1) lookups for retrieving flight data.
- **2.2.3** Compare the execution times of hash-based lookup and binary search to evaluate performance on large datasets and query volumes.

Binary Search Tree (BST) Queries

- **2.3.1** Store flight records using a Binary Search Tree (BST) for efficient insertion, searching, and traversal.
- **2.3.2** Analyze flight data

Task 2.1 - Linear vs Binary Search



Binary Search:

The mean execution time (y-axis, in microseconds) remains low and stable across all dataset sizes (x-axis, from 10 to 100,000 IDs) for small, medium, and large datasets. Confirms the logarithmic time complexity **(O(log n))** of binary search

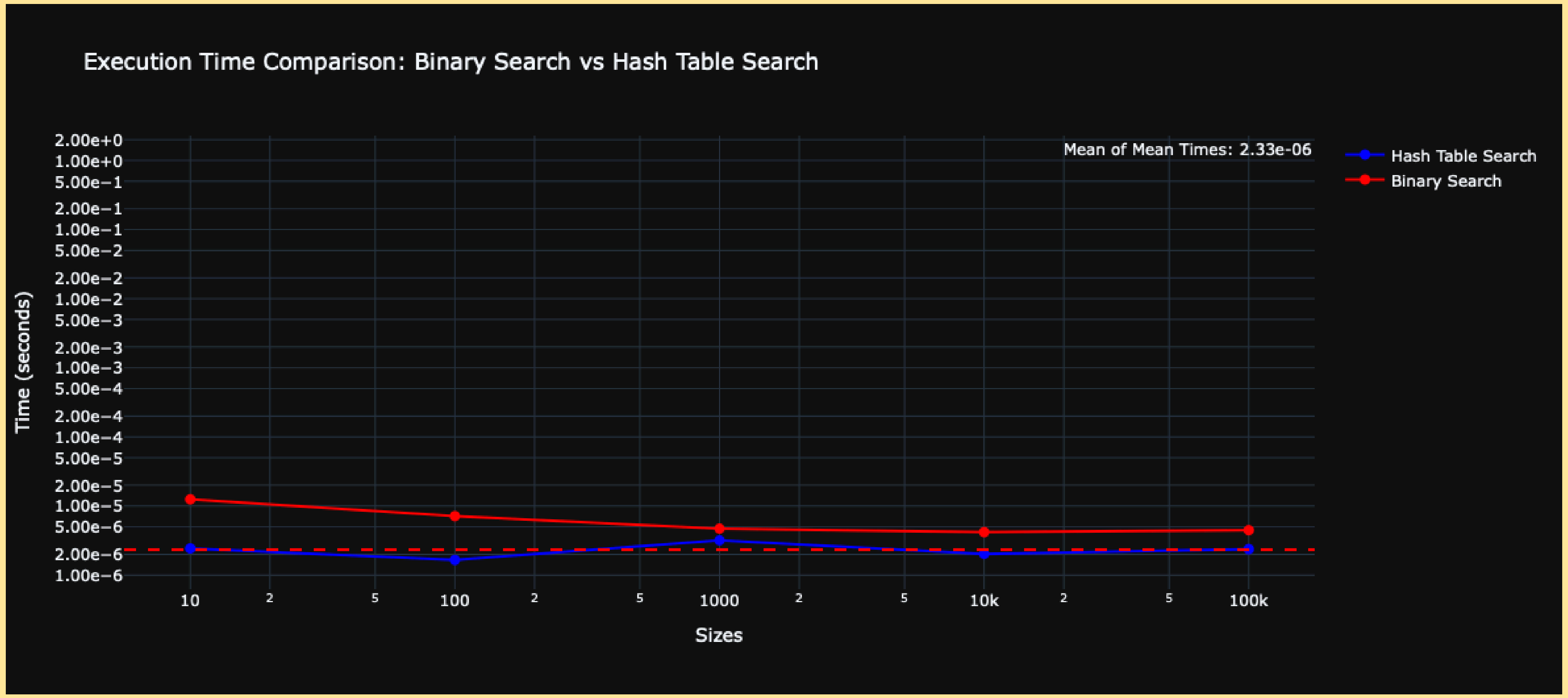
Linear Search:

The mean execution time (y-axis, in seconds) increases significantly for the "Large Dataset" (green line) as the number of test IDs grows. Reflects the linear time complexity **(O(n))** of linear search.

Conclusion:

The experimental results highlight the superior performance of binary search over linear search for sorted data. Binary search shows low and stable execution times, maintaining efficiency even with large datasets due to its logarithmic scaling. In contrast, linear search becomes inefficient as dataset size increases, leading to longer execution times.

Task 2.2 - Hash Table Lookup vs binary search



<u>Hash Table Search:</u>

- The execution time remains constant and very low across all tested data sizes (from 10 to 100,000).
- Supports the expected **O(1) average-case time complexity** for hash table lookups. The search time is virtually independent of the dataset size.
- The horizontal dashed blue line represents the mean execution time for hash table search (2.30e-06 seconds), highlighting this consistent performance.

<u>Binary Search:</u>

- The execution time shows a gradual increase as the data size grows.
- O**(log n) time complexity** of binary search. Time increases, at a logarithmic rate, which is still significantly better than linear search.

2.3 - Binary Search Tree (BST) Queries

2.3.1 Store flight records using a Binary Search Tree (BST) for efficient insertion, searching, and traversal.
2.3.2 Analyze flight data to find:

Fast Operations (Avg. O(log n)) - Efficient insertion, search, and traversal thanks to the tree structure
Sorted Access - In-order traversal retrieves flights in ascending flight number.
Range Queries - Quickly find all flights that fall within a specific range of flight numbers.

Operation	**Time Complexity**	**Description**
Insert	O(log n)	Keeps tree sorted
Search by Flight Number	O(log n)	Finds flights quickly
In-Order Traversal	O(n)	Lists flights in order
Most Frequent Airline	O(1) insert, O(k) query	Tracks and retrieves most common airline
Flights from Origin	O(1) insert & query	Quickly filters by origin
Longest Delays	O(1) insert, O(n log n) query	Can be optimized further
Flight Numbers in Range	O(log n + k)	Retrieves flights in number interval
Avg. Delay per Airline	O(1) update, O(k) query	Constant-time updates, fast queries
Busiest Destination	O(n) traversal, O(1) count	Tracks per node counts
Flights Departed Early	Optimized traversal	Stops early if condition met

TASK 3-Graphs and Connected Components

Graph Modeling and Visualization

- **3.1.1** Import flight data and use NetworkX to create a graph.
- **3.1.2** Perform a representation of each airport as a node in the graph.
- **3.1.3** Generate edges between nodes if there is at least one connecting flight between the two airports.
- **3.1.4** Assign edge weights based on the number of flights between each pair of airports.
- **3.1.5** Generate the graph using Matplotlib and NetworkX, including node labels and edge weights

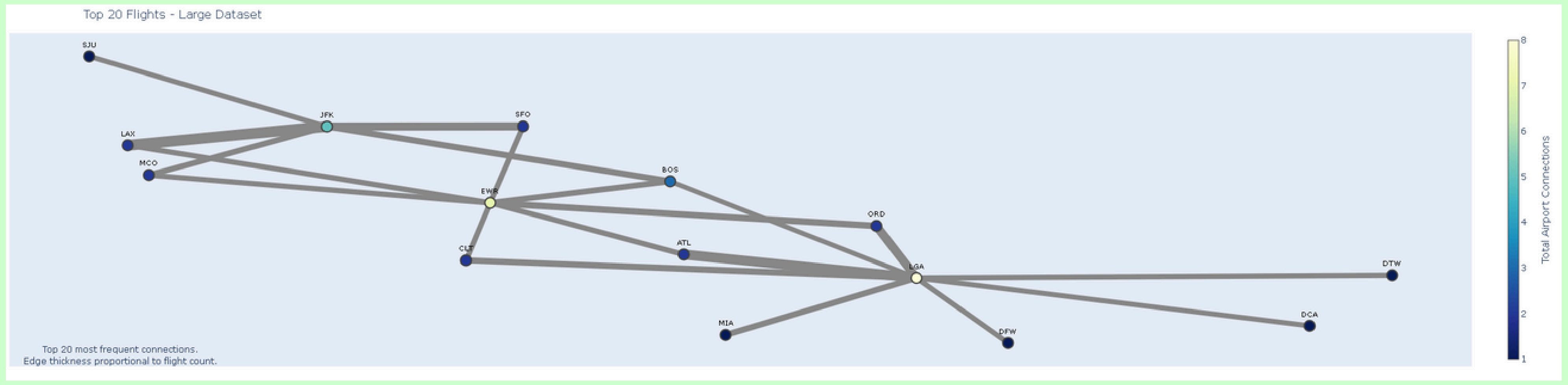
Connected Components Detection

- **3.2.1** Consider your undirected graph for traversal in both directions, revealing intricate relationships.
- **3.2.2 Level 1:** Use the networkx.connected_components() function to find the connected components in your graph.
- **3.2.3 Level 2:** Implement manually either the Depth-First Search (DFS) or Breadth-First Search (BFS) algorithm to find the connected components, through traversal techniques.
- **3.2.4** Compare the networkx and manual methods' results to validate the findings and check the accuracy.
- **3.2.5** Visualize or print each discovered component to enhance comprehension and create the graph's connections

DFS, BFS & Performance Comparison

- **3.3.1** Measure the time and memory usage of the execution of DFS and BFS, on a randomly generated graph
- **3.3.2** Visualize and compare their performance to find connected components on graphs of different sizes.

Task 3.1 & Task 3.2 - Graph Modeling and Visualization



Graph Modeling 3.1

- **Nodes**: Represent individual airports (107 distinct airports).
- **Edges**: Indicate direct flight connections between airports.
- **Edge Weight:** Thickness proportional to the number of flights between the connected airports.
- **Node Color:** Lighter colors signify airports with the highest number of connections within the top 20 routes.

Key Insights:

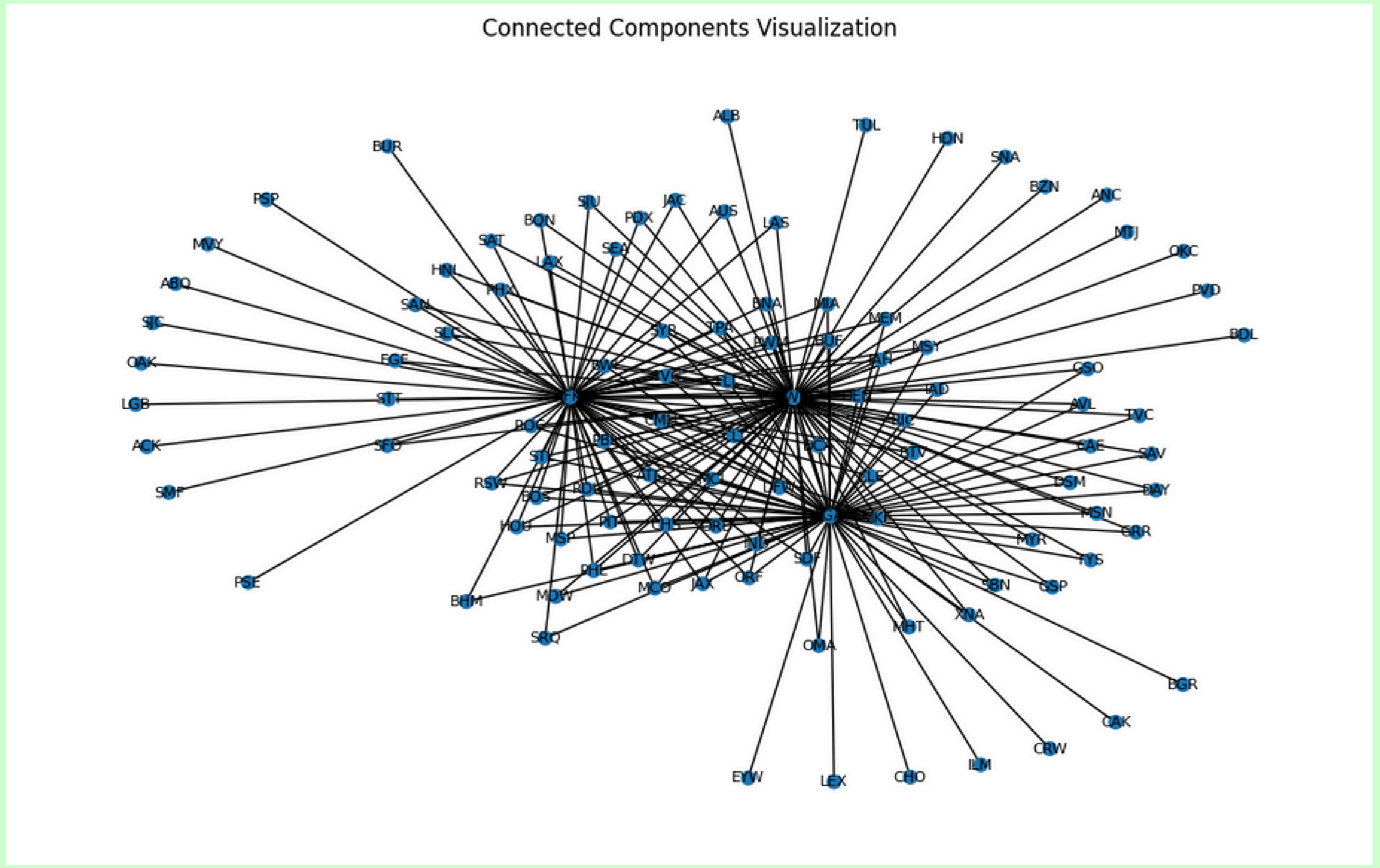
- Identifies highly connected "hub" airports (lighter nodes).
- Highlights the busiest flight routes (thicker edges).
- Visualizes the structure of the most significant air traffic network.

Connected Components (3.2)

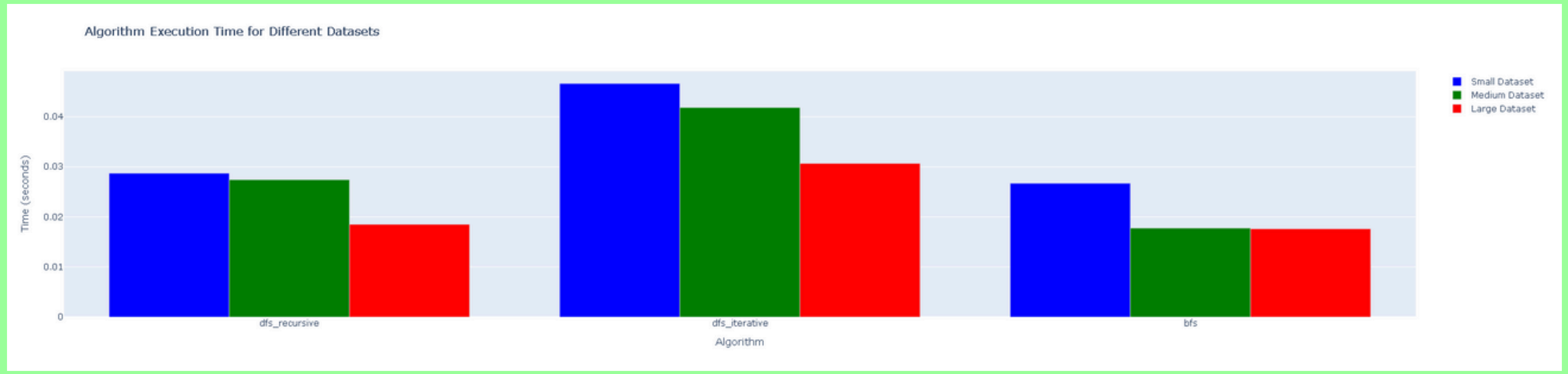
Graph Type: Undirected Graph
Results: Single Connected Component

Network Structure:

Dense central cluster = major hub airports (e.g., ATL, DFW, ORD).
Peripheral nodes = smaller, regional airports with fewer direct connections



Task 3.3 – DFS & BFS & Performance Comparison:

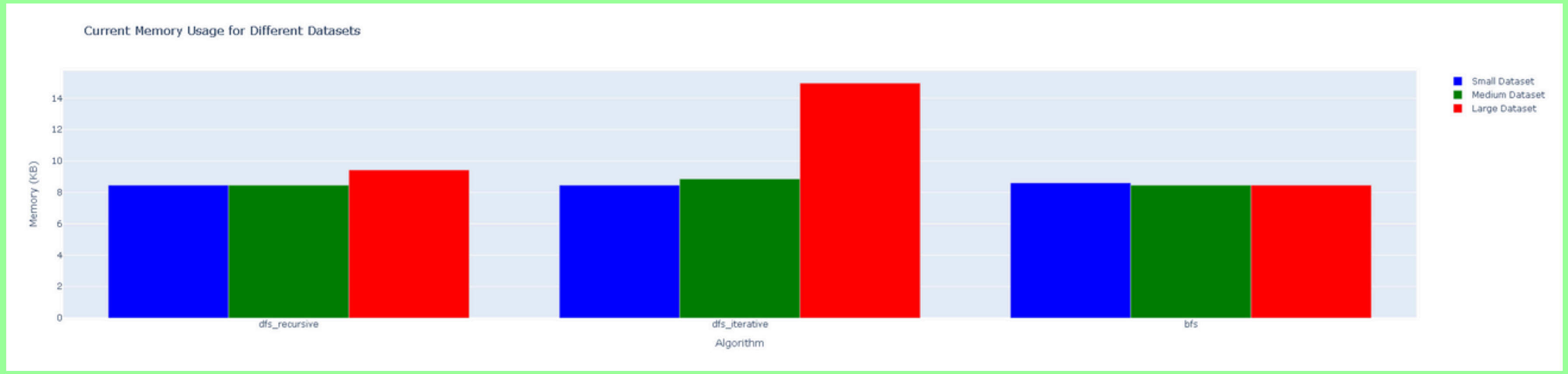


Execution Time Analysis:

BFS (rightmost bars): Consistently the fastest across all dataset sizes (small, medium, and large).

Recursive DFS (leftmost bars): Slightly faster than iterative DFS, especially for larger datasets.

Iterative DFS (middle bars): The slowest in all scenarios, likely due to stack management overhead.



Memory Usage (Peak):

Recursive DFS (blue): Lowest peak memory usage (~10.45 KB).

BFS (red): Low memory footprint (~11.77 KB)

Iterative DFS (green): Highest memory usage (~444 - 447 KB)

Conclusions:

Best for speed: BFS is the optimal choice.

Best for memory: Recursive DFS is the most efficient, but be mindful of potential stack overflow in very deep graphs.

Least Efficient: Iterative DFS, high memory without offering speed benefits.

Scalability: Recursive DFS and BFS demonstrate better memory scalability compared to iterative DFS.

CONCLUSION

11 May, 2025