Final Project Report

1. <u>**Overview of the Data**</u>

   The dataset, *Degrees That Pay Back*, contains financial metrics for various undergraduate degrees. Key attributes include Undergraduate Major, Starting Median Salary, Mid-Career Median Salary, and additional percentile data for salaries at different career stages (e.g., 10th, 25th, 75th, 90th). The analysis focuses on the **Mid-Career Median Salary**, which provides a measure of financial outcomes for professionals in their mid-career.

2. <u>**Chosen Data Structures**</u>

   Hash Table: The hash table was used to map each undergraduate degree to its mid-career median salary. This structure allows for efficient insertion and retrieval operations, making it ideal for scenarios where data does not need to be stored in any particular order. The average time complexity for these operations is O(1), which ensures fast performance even for larger datasets.

   Binary Search Tree: A custom implementation of a BST was employed to store degrees based on their salary values in an ordered manner. The hierarchical structure of the BST allows for ordered traversal, making it easy to retrieve the maximum value by traversing to the rightmost node. The average time complexity for operations like insertion and search is O(log n), assuming the tree remains balanced.

3. <u>**Question Being Answered**</u>

   Which undergraduate degree offers the highest mid-career median salary?

   This question is significant because salary data is a critical factor for students choosing a major and for policymakers analyzing the economic impacts of educational investments. Identifying degrees with the highest financial returns provides valuable information for career planning, institutional decision-making, and educational funding strategies. The question requires efficient data organization and analysis due to the need for maximum value extraction from a dataset.

4. <u>**Data Cleaning Steps**</u>

Currency Formatting: Salary fields initially contained symbols (e.g., $) and commas, which were removed to convert them into numerical format suitable for computation.

Missing Values: Rows with missing or invalid entries in the Mid-Career Median Salary column were dropped to ensure the reliability of the analysis.

Column Selection: Only the Undergraduate Major and Mid-Career Median Salary columns were retained for analysis to focus on the data relevant to the research question.

Numeric Validation: Ensured all salary values were positive and correctly formatted for mathematical operations.

These cleaning steps addressed common data quality issues, making the dataset suitable for the analysis performed using both data structures.

5. Operations and Implementations

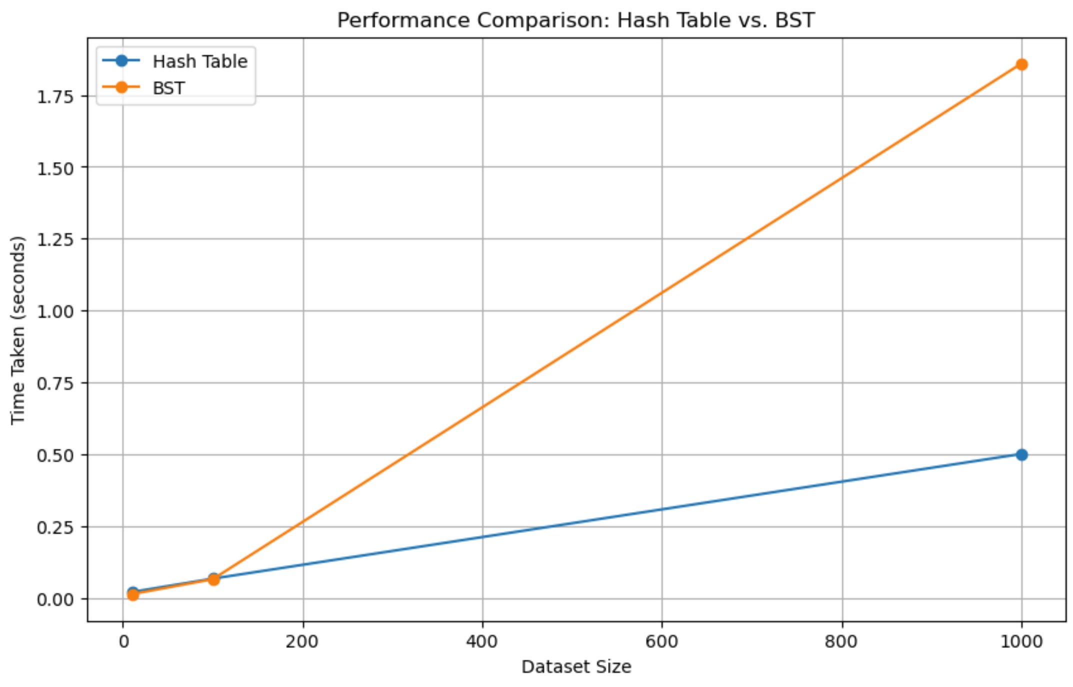
Hash Table Operations: In the hash table implementation, the Undergraduate Major was used as the key, and the Mid-Career Median Salary as the value. The insertion operation populated the hash table with degree-salary pairs, and the retrieval operation iterated through the hash table to find the maximum salary. This approach ensured efficient data storage and retrieval, with an average time complexity of **O(1)** for insertion and lookup operations.

Binary Search Tree Operations: In the BST implementation, each degree and its corresponding salary were stored in nodes, ordered by salary values. The insertion operation placed nodes in the correct position based on salary, maintaining the tree's sorted structure. Retrieving the maximum salary involved traversing to the rightmost node, as it contained the highest value in a BST. The average time complexity for insertion and retrieval was **O(log n)** for a balanced tree.

6. Performance Analysis

Hash Table Performance: The hash table demonstrated exceptional efficiency for this analysis, with an average time complexity of O(1) for insertion and retrieval operations. It stored undergraduate majors as keys and their corresponding mid-career median salaries as values, allowing for rapid data access and straightforward traversal to find the maximum salary. The hash table consistently maintained fast performance even as dataset size increased, as shown in the graph, making it an optimal choice for this specific task where order and range queries were not required.

BST Performance: The Binary Search Tree (BST) offered a structured way to store data in an ordered manner, with an average time complexity of O(log n) for balanced trees during insertion and search. It was implemented to store majors and salaries in nodes ordered by salary, enabling efficient maximum salary retrieval by traversing to the rightmost node. However, the BST's runtime grew significantly as dataset size increased, particularly if the tree was unbalanced, making it slower than the hash table for large datasets. Despite its ordering advantages, the BST was less suited to this analysis where speed was paramount.



For this project, where the goal was simply to find the maximum mid-career median salary, the Hash Table proved to be the superior data structure due to its minimal runtime and simplicity. The Binary Search Tree offers advantages in scenarios requiring ordered data but was less efficient for the task at hand. The final result, identifying **Chemical Engineering** as the degree with the highest mid-career median salary at **$107,000**, was achieved efficiently using both data structures. However, the **hash table** achieved the result faster.