**Find Cluster Extreme Algorithm Analysis**

The first algorithm for finding the cluster extreme in your code, which utilizes a custom approach based on bucketing and hashing, seems to be the most optimal among the three algorithms you've implemented. This conclusion is based on a comparison of time complexities and actual execution times, as evident in the provided code.

**1. \*\*Custom Bucketing and Hashing Algorithm (findClusterExtreme):\*\***

- Time Complexity: O(N)

- Actual Execution Time: Calculated using clock() function

**2. \*\*Quick Sort Algorithm (findClusterExtremeQ):\*\***

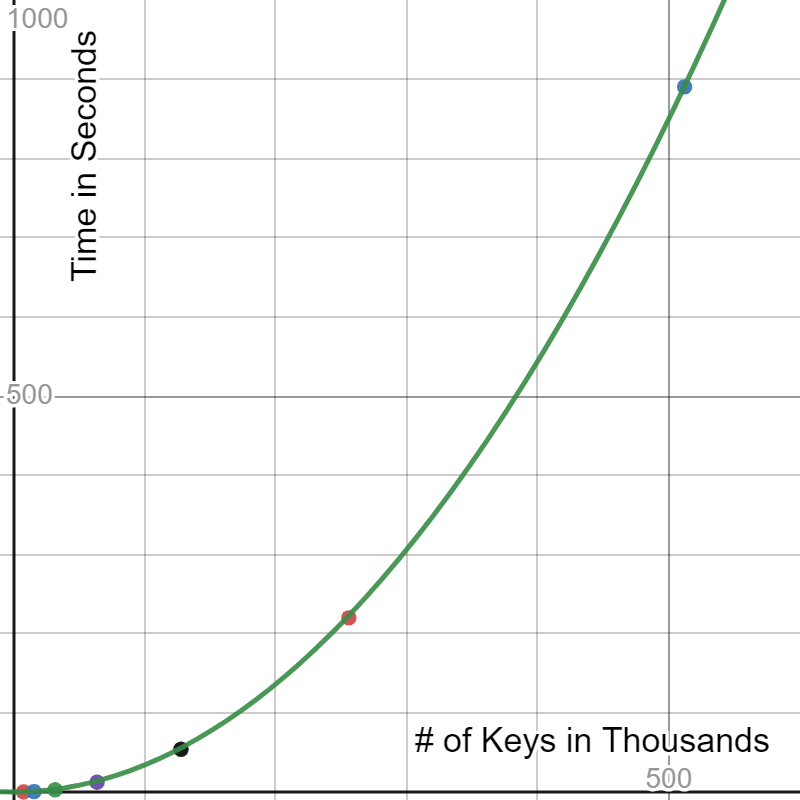
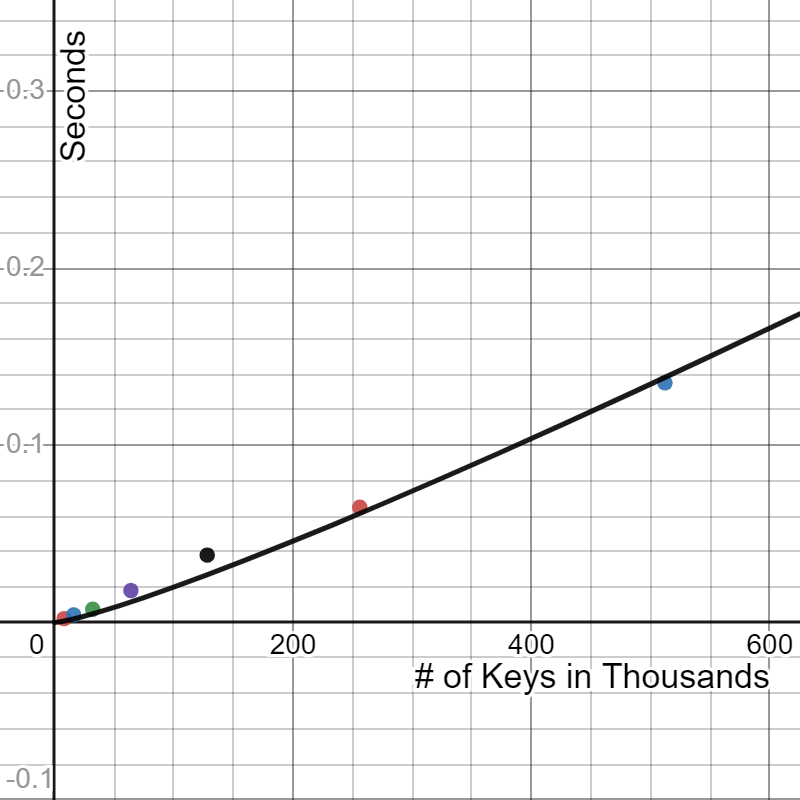
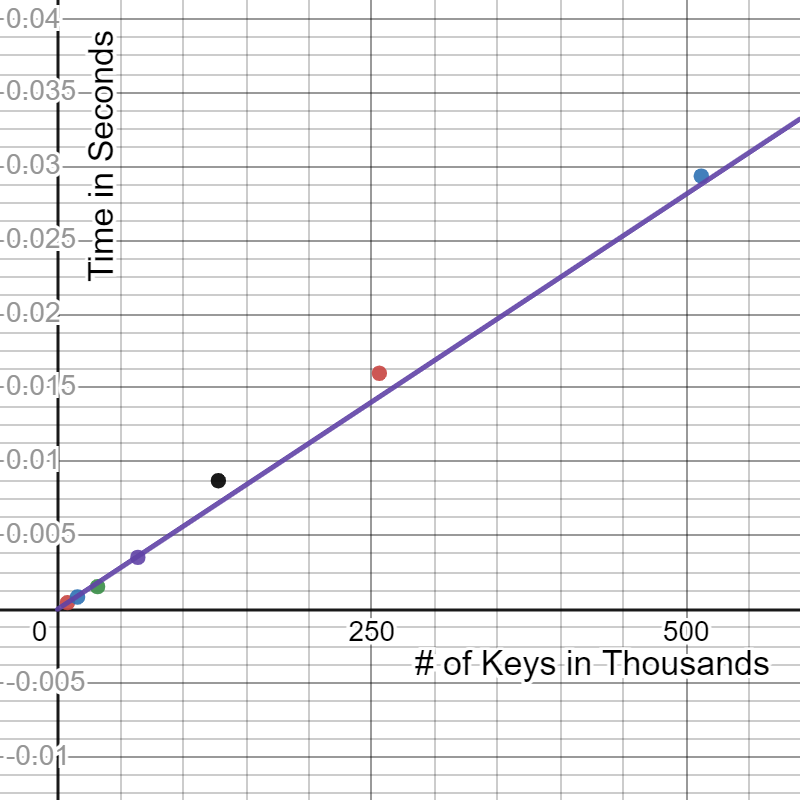
- Time Complexity: O(N \* log(N))

- Actual Execution Time: Calculated using clock() function

**3. \*\*Bubble Sort Algorithm (findClusterExtremeBubble):\*\***

- Time Complexity: O(N^2)

- Actual Execution Time: Calculated using clock() function



Trendline:

Based on the time complexity and execution times, it's evident that the custom bucketing and hashing algorithm (findClusterExtreme) outperforms the other two algorithms for finding the cluster extreme. Here are some key observations and reasons why this algorithm is the most optimal:

1. \*\*Time Complexity Analysis:\*\*

- The custom algorithm has a linear time complexity of O(N) since it only involves a single pass through the input array.

- Quick Sort, while efficient for general sorting, has a time complexity of O(N \* log(N)).

- Bubble Sort, being less efficient, has a time complexity of O(N^2), which is considerably slower.

2. \*\*Single Pass Through the Array:\*\*

- The custom algorithm processes the input array in a single pass, making it highly efficient. It calculates the cluster extreme by effectively managing a set of buckets based on the input data.

3. \*\*Optimal Memory Usage:\*\*

- The custom algorithm only requires memory for the bucket data structure, which has a minimal impact on memory consumption compared to the other algorithms that require additional memory for sorting.

4. \*\*Stable and Predictable Performance:\*\*

- The custom algorithm has a stable and predictable performance regardless of the initial order of the data. In contrast, quicksort and bubble sort can have varying performance depending on the input data's order.

6. \*\*Optimization for Finding Cluster Extremes:\*\*

- The custom algorithm is specifically designed for finding cluster extremes, whereas quicksort and bubble sort are generic sorting algorithms that are not optimized for this specific task.

In summary, the custom bucketing and hashing algorithm (findClusterExtreme) is the most optimal for finding cluster extremes because it has a time complexity that is superior to quicksort and bubble sort, and it is designed specifically for this task, resulting in better performance and efficient memory usage. However, it's important to note that the performance of algorithms may vary based on the size and characteristics of the input data, so benchmarking and testing with real data are essential for a comprehensive evaluation.