Assignment 2

Report on different algorithms to find Minimum and Maximum elements in a array of n size.

SAYAK SEN

2023CSB047

1.Analysis : Finding Min and Max Element

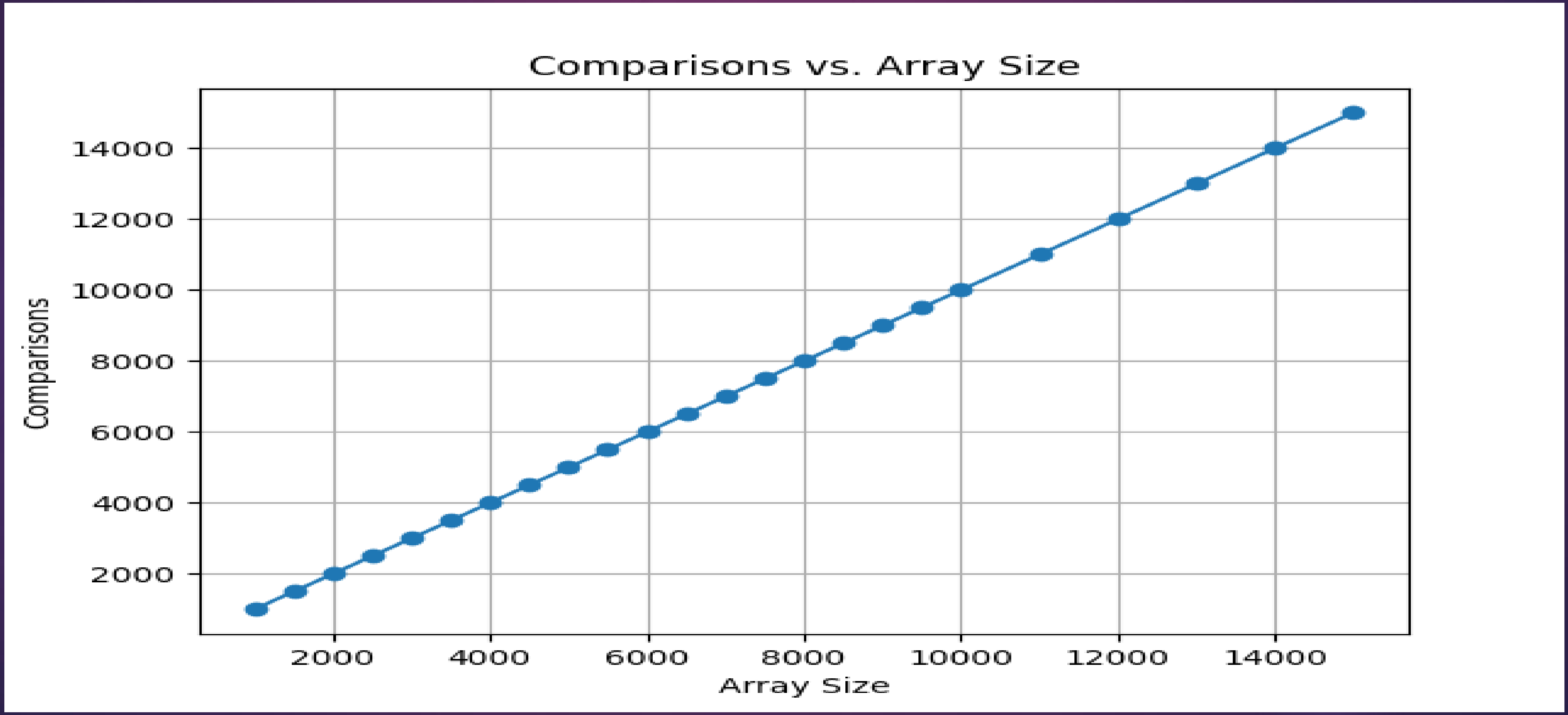
1. Naive Approach
The naive approach iterates through the array once, keeping track of the current minimum and maximum values encountered so far. It initializes both the minimum and maximum to the first element of the array. Then, for each subsequent element, it compares the element to the current minimum and maximum, updating them if necessary.

•**Runtime Complexity:** The naive approach has a time complexity of O(n), where n is the number of elements in the array. This is because it iterates through the array once.

•**Advantages:** Simple to understand and implement.

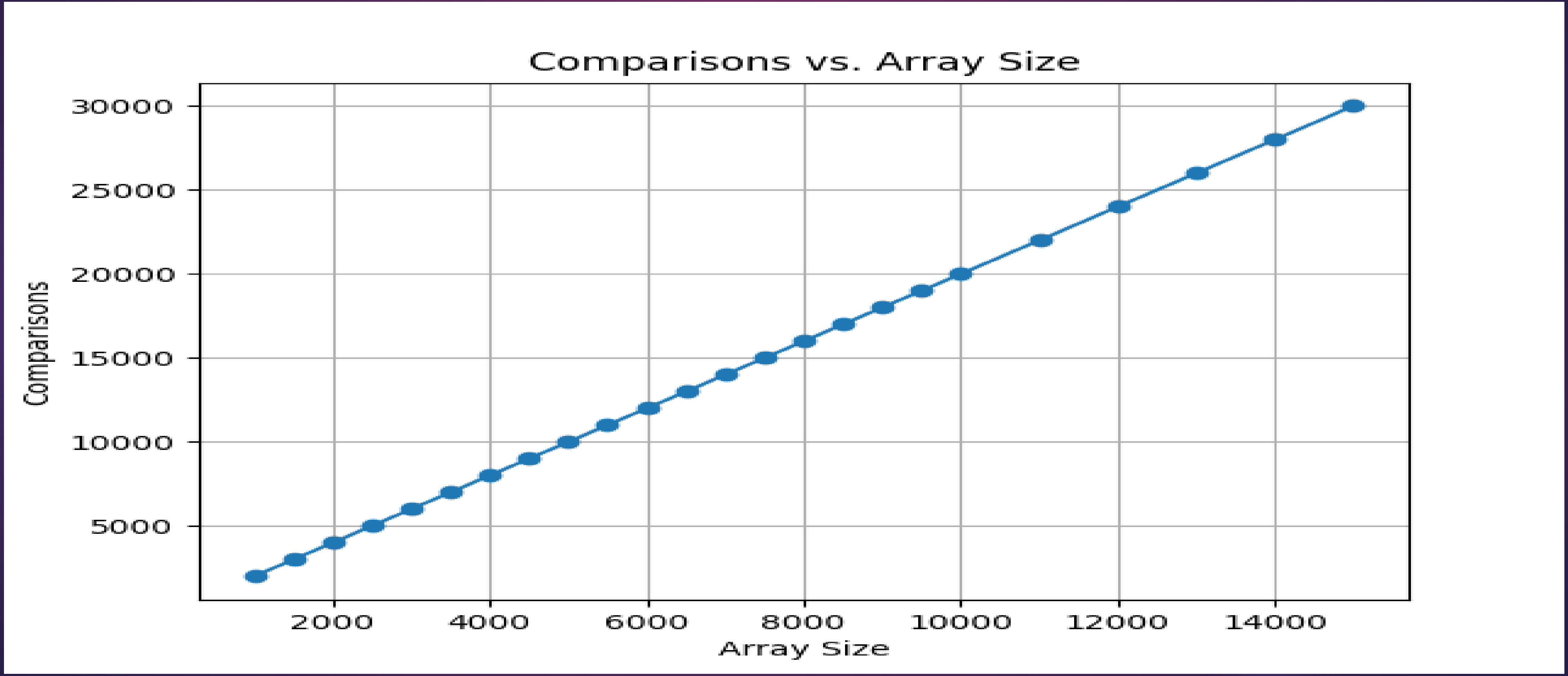
•**Disadvantages:** While linear time is efficient, there might be algorithms that can theoretically achieve the same result with fewer comparisons.

1. Naïve Algorithm to find min and max in array
Comparisons = 2n-2



- **2. Divide-and-Conquer Approach**
- The divide-and-conquer approach recursively divides the array into smaller subproblems until the subproblems become trivial (i.e., containing one or two elements). It then combines the results from the subproblems to find the overall minimum and maximum.
  - **Runtime Complexity:** The divide-and-conquer approach also has a time complexity of O(n). While the recursive calls might seem complex, the total number of comparisons made is still proportional to n.
  - **Advantages:** Can be more efficient in practice for certain types of data or hardware architectures due to better cache utilization or parallelization possibilities (though the basic algorithm as described is not parallel). Provides a different perspective on the problem.
  - **Disadvantages:** More complex to implement than the naive approach due to the recursion. The overhead of recursion can sometimes make it slower in practice for smaller datasets.
- **3. Comparison**
- Both the naive and divide-and-conquer approaches have the same time complexity of O(n). In most practical scenarios, the naive approach is often preferred due to its simplicity and lower overhead. The divide-and-conquer approach might offer some performance benefits for very large datasets or in specific hardware environments, but its increased complexity often outweighs these potential benefits in simpler cases.
- **4. Conclusion**
- Finding the minimum and maximum values in an array is a fundamental problem in computer science. While multiple algorithms exist, the naive and divide-and-conquer approaches are two common and effective methods. The naive approach is usually the most practical choice due to its ease of implementation and comparable performance. The divide-and-conquer approach offers an alternative perspective and could be more efficient in specialized situations.

Divide and Conquer to find min and max element in array Comparisons = 2n-2



Analysis : Finding Min and Second Min in an array

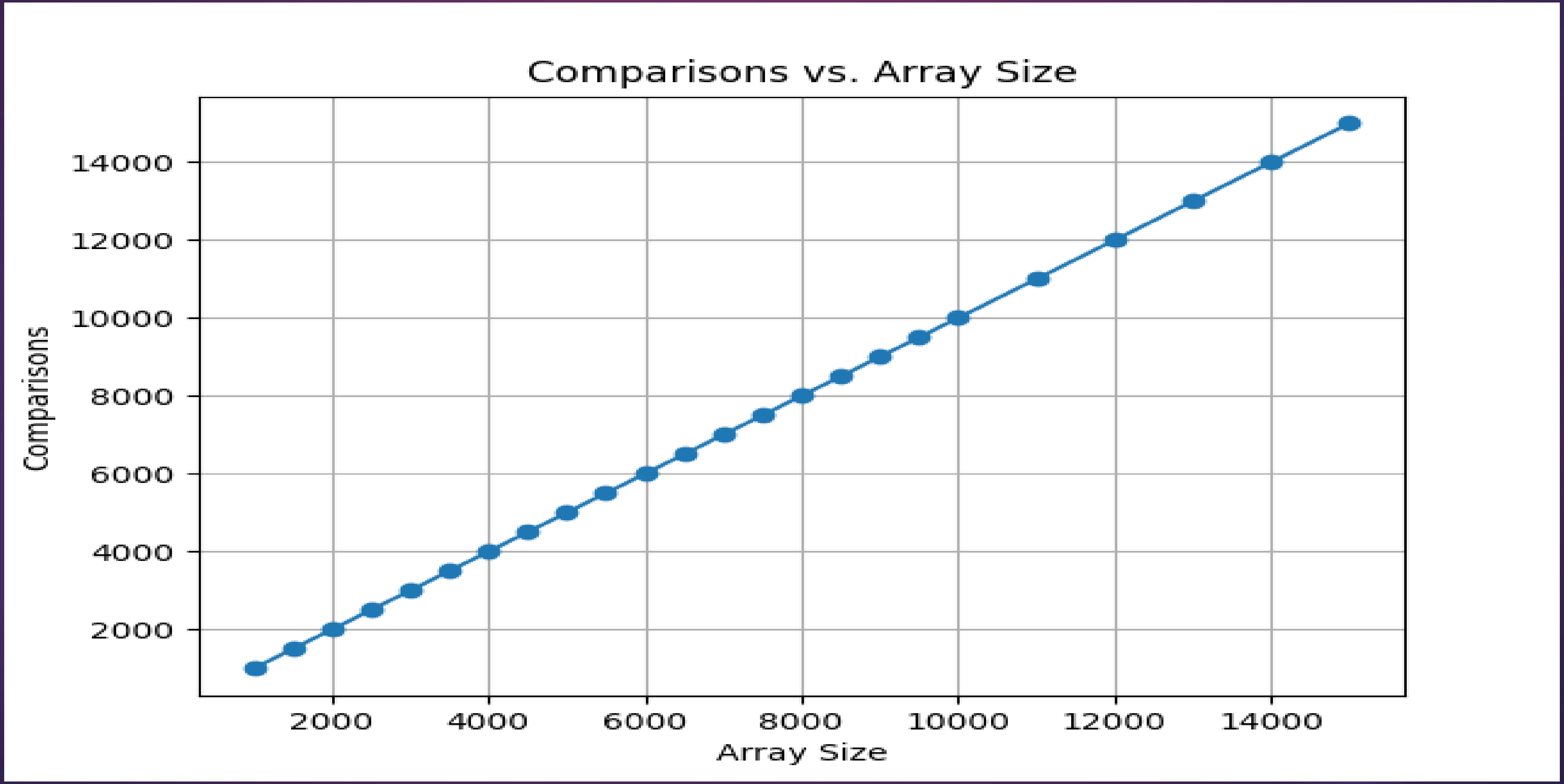
1. Naive Approach
The naive approach iterates through the array, keeping track of the current minimum and second minimum values encountered so far. It initializes the minimum and second minimum to the first two elements (handling edge cases where the array has fewer than two elements). Then, for each subsequent element, it compares the element to the current minimum and second minimum, updating them if necessary.

•**Runtime Complexity:** O(n), where n is the number of elements in the array, due to a single pass through the data.

•**Advantages:** Simple to understand and implement.

•**Disadvantages:** Can potentially perform more comparisons than necessary.

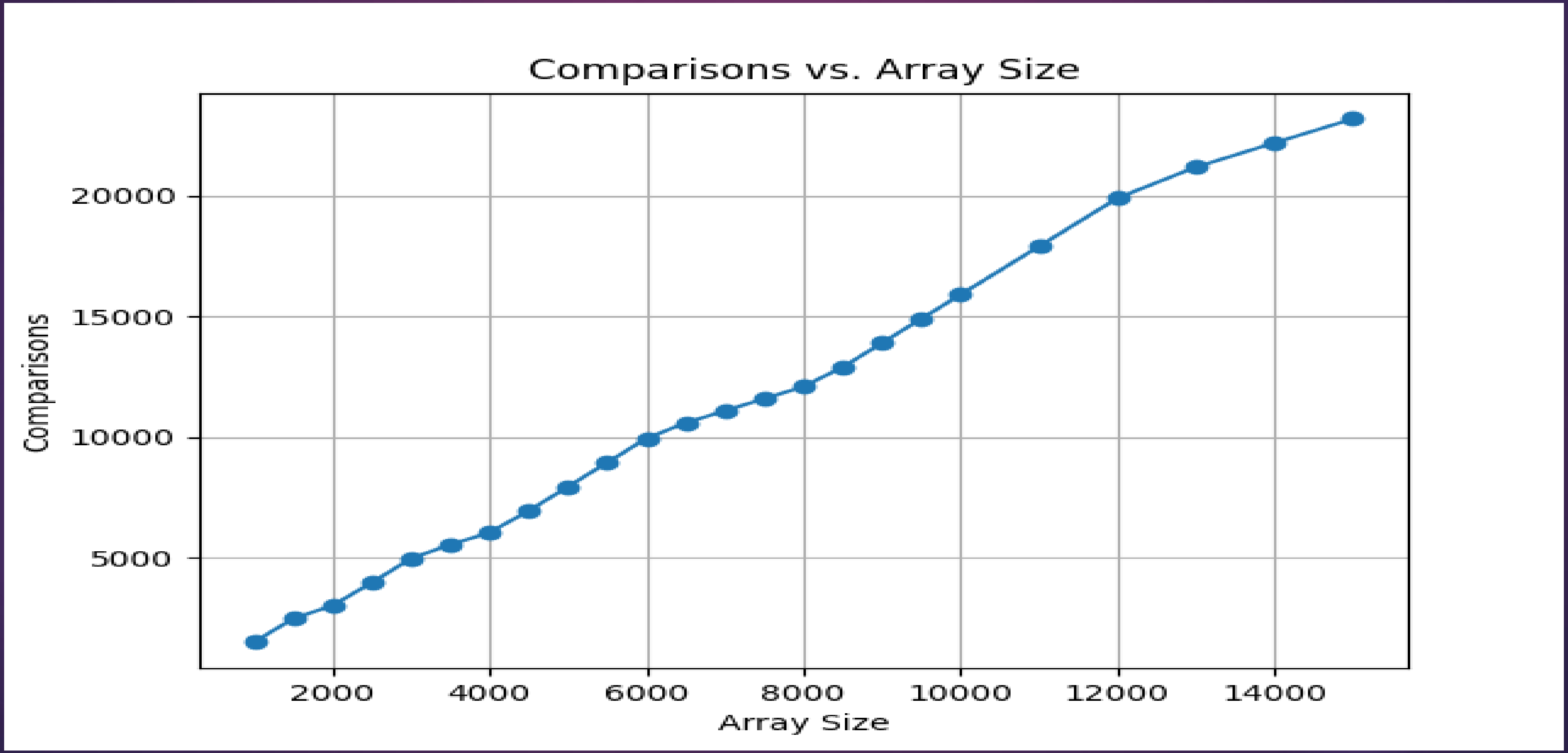
2.Naïve Algorithm to find min and second min in array Comparisons = n-1



- **2. Divide-and-Conquer Approach**
- The divide-and-conquer approach recursively divides the array into smaller subproblems. The base cases handle arrays of size 1 or 2. The results from the subproblems are then combined.

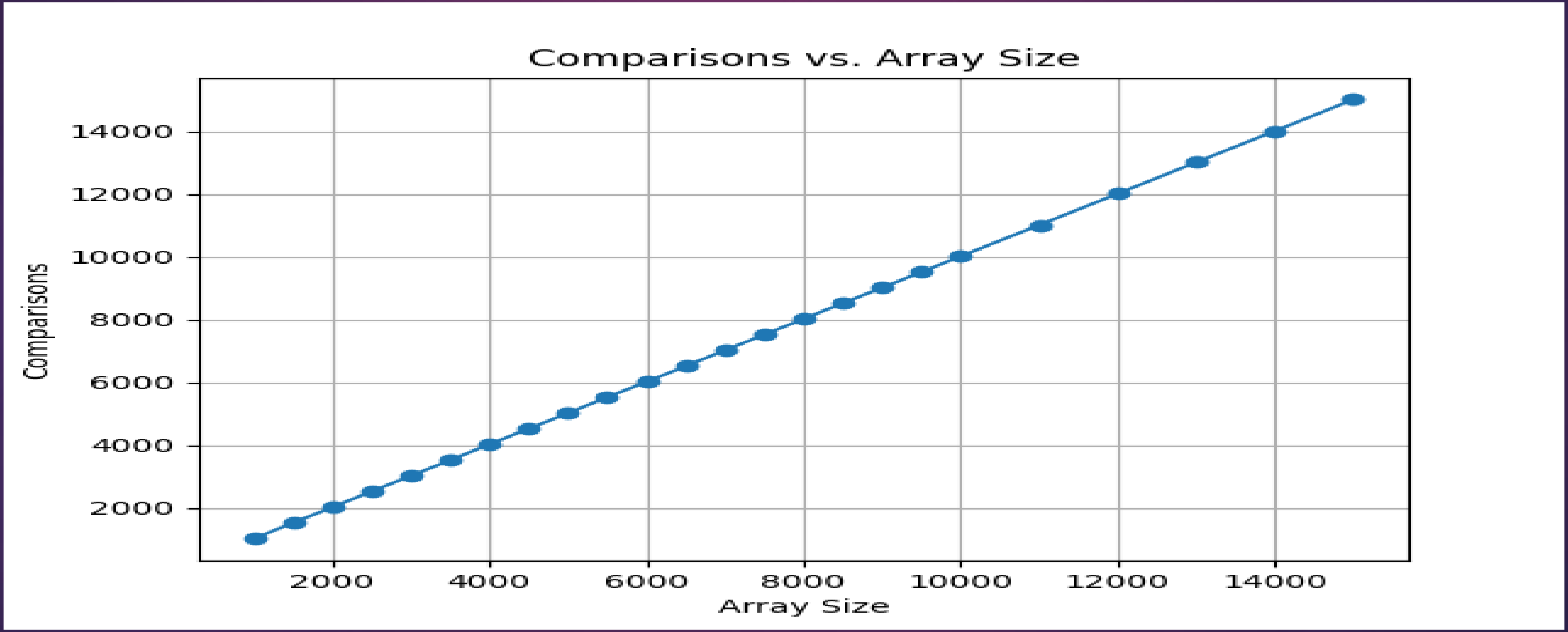
- **Runtime Complexity:** O(n). The recurrence relation is similar to merge sort, leading to linear time complexity.
- **Advantages:** Provides a different algorithmic approach.
- **Disadvantages:** More complex to implement due to recursion. The overhead of recursion can be significant for smaller datasets.

Divide and Conquer to find min and second min in array Comparisons = n-1



- **3. Tournament Method**
- The tournament method mimics a single-elimination tournament. It pairs elements and compares them, with the "winners" (smaller elements) advancing. The overall winner is the minimum. The second minimum must have "lost" to the minimum at some point.
  - **Runtime Complexity:** Building the tournament tree takes O(n) comparisons. Finding the second minimum by traversing the path also takes O(log n) comparisons (the height of the tree). The overall complexity is dominated by the tree building, making it O(n).
  - **Advantages:** Can be efficient in practice, particularly if you need to find the k-th smallest element (not just the second smallest).
  - **Disadvantages:** More complex to implement than the naive approach. Requires building and storing the tournament tree.
- **4. Comparison**
- All three methods have a time complexity of O(n). In practice, the naive approach is often the simplest and most efficient for finding the minimum and second minimum. The tournament method and divide-and-conquer method are more complex to implement. The tournament method can be advantageous if you need to find the k-th smallest element, as it provides a basis for more general selection algorithms. The divide-and-conquer method, while offering a different perspective, usually doesn't provide performance benefits in this specific scenario.

Tournament Method to find min and second min in array
Comparisons = n-1 + logn


Comparisons vs. Array Size
Comparisons
Array Size