1. Does the runtime grow linearly as input number n increases in LinearFibonacci.java?

   Yes, the runtime of the LinearFibonacci algorithm grows linearly as the input number n increases. This is because the algorithm has a time complexity of O(n), meaning the runtime increases linearly with the size of the input.

2. Does the runtime grow exponentially as the input number n increases in ExponentialFibonacci.java?

   Yes, the runtime of the ExponentialFibonacci algorithm grows exponentially as the input number n increases. This is due to the recursive nature of the algorithm, which results in repeated calculations of subproblems, leading to exponential growth in runtime. The time complexity of the ExponentialFibonacci algorithm is O(2^n).

3. How often is the linear algorithm faster than the exponential algorithm in ExponentialFibonacci.java on the same input?

   The linear algorithm (LinearFibonacci) is significantly faster than the exponential algorithm (ExponentialFibonacci) on the same input, especially for larger values of n. The difference in runtime becomes more pronounced as n increases due to the exponential growth of the ExponentialFibonacci algorithm's runtime.

4. Write a report that describes your work done in the following sections:

   Analyzing the runtime behavior of the LinearFibonaccia and ExponentialFibonacci to discover how the runtime of each algorithm scales with the input. Using algorithms in Java and analysing runtime using R. Comparing the runtime for various input various of n and compared their performance. Revealing that the runtime of the LinearFibonacci algorithm grows linearly with the input of n size. The ExponentialFibonacci algortithm exibited exponential growth in runtime as n increased. Confirming the analysis of the algorithms' time complexities. With the differences highlighting the imortance of algorithm efficiency. The LinearFibonacci algorithm outperforms the ExonentialFibonacci Algorithm, especially for larger input values. Meaining the LinearFibonacci would be prefererd due to its better uses in scalability and efficiency. In conclusion, our evaluation shows the significance of algorithmic complexity on runtime performance.