How to Run

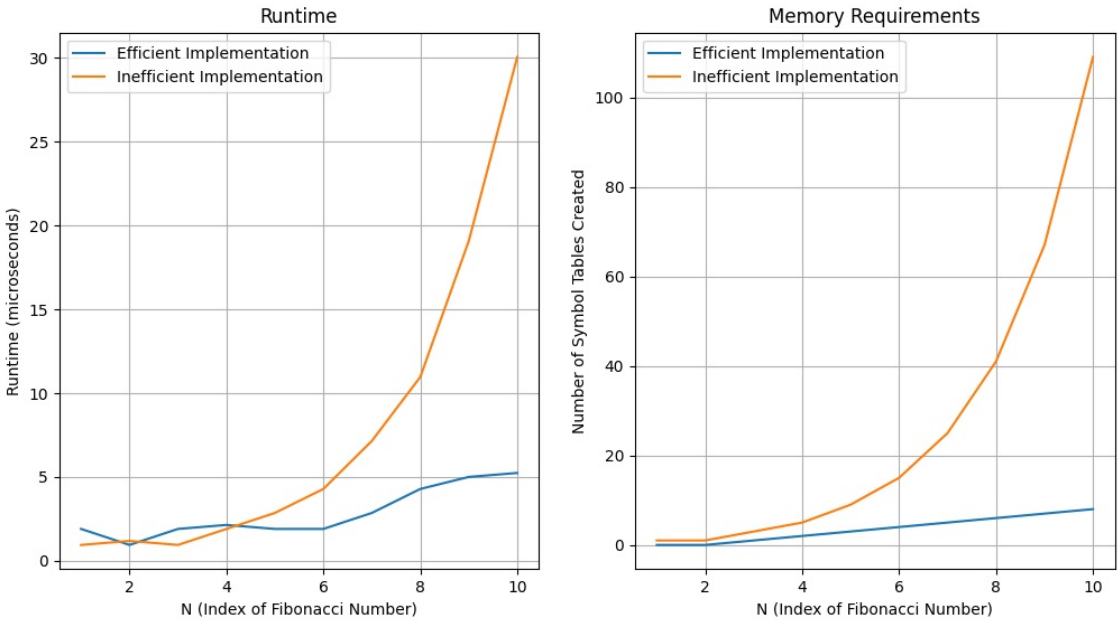
To run this program from the terminal, enter `python Lab9.py`.

Inputs

The program does not require inputs to run, however inputs could be implemented to allow the user to select the desired number of Fibonacci numbers.

Outputs

The program generates a graph with two subplots. The first subplot displays the runtime for efficient and inefficient Fibonacci calculators as the index of the Fibonacci sequence increases. The second subplot displays the memory requirements for the efficient and inefficient calculators.



Findings

Storing the calculated Fibonacci numbers in a dictionary allowed the efficient implementation to outperform the inefficient implementation. By storing the calculated values, this cut down on the number of recursive calls made by fib_efficient. On the other hand, fib_inefficient performed much slower and had greater memory requirements because it had to recalculate the previous Fibonacci number multiple times within a single trial.

Concept Map

No concept map was assigned for this program.

Collaborators

Nisa Genc found an article on Stack Overflow that suggested the use of memoization during our team brainstorming meeting.

Devon Gardner helped me to debug my code when I ran into some problems with the symbol table counter.

References

Stack Overflow. (2013, Aug 11.) *Efficient Calculation of Fibonacci Series*. https://stackoverflow.com/questions/18172257/efficient-calculation-of-fibonacci-series

> This source served as the inspiration to implement memoization.

Tech With Tim. (2017, Oct 8.) *Recursion and Memoization Tutorial Python*. https://www.youtube.com/watch?v=sCecRPSQg6Y

> This source helped me to actually implement memoization.