Report: Carmichael Function with Performance Analysis

AIM

To implement the Carmichael function λ(n) using prime factorization and mathematical properties.

To compute λ(n) for any positive integer n by finding the least common multiple of Carmichael values for each prime power factor.

To measure and analyze performance metrics (execution time and memory usage) for the computation.

To visualize the behavior of λ(n) across a range of values using matplotlib for better understanding of the function's growth pattern.

METHODOLOGY & TOOLS USED

Language: Python 3

Libraries & Tools: - math – Mathematical operations - time – Execution time measurement - os – Operating system interface - psutil – Process memory monitoring - matplotlib.pyplot – Data visualization and graphing

Key Algorithms:. Prime Factorization: Using trial division up to √n to identify factors and their exponents. GCD (Greatest Common Divisor): Utilizing the Euclidean algorithm to calculate gcd(a, b). LCM (Least Common Multiple): Determined via the formula LCM(a,b) = (a × b) / GCD(a,b). Carmichael Prime Power:. When p = 2 and k ≥ 3: λ(2^k) = 2^(k-2). For odd primes: λ(p^k) = (p-1) × p^(k-1) - LCM Combination: Final result is the LCM of all λ(p^k) values

BRIEF DESCRIPTION

The program determines the Carmichael function λ(n) by performing these steps:

Prime Factorization (primefactorspowers(n)): Decomposes n into its components along, with their corresponding exponents, represented as tuples (p, k).

Carmichael, for Prime Powers (carmichaelprimepower(p, k)): Utilizes the expression to determine λ(p^k) depending on whether p equals 2 or is an odd prime.

Main Computation (carmichael(n)):

Handles the special case n=1 (returns 1)

Collects λ values for each prime power factor

Computes the LCM of all these values to get λ(n)

Performance Tracking:

Records execution time using time.time()

Measures memory usage in bytes via psutil.Process().memory_info().rss

Visualization:

Computes λ(n) for n = 1 to 50

Plots results with markers using matplotlib

Saves the plot as carmichael_lambda_plot.png

RESULTS ACHIEVED

Correctness: The software precisely calculates the Carmichael function across input values

Output Example: For a given user input n, the program displays:

The computed λ(n) value

Memory usage in bytes

Execution time in seconds (to 6 decimal places)

Graph Display: A illustration depicting the variation of λ(n) from 1 to 50 highlighting unique trends, for prime powers, composite integers and exceptional instances

Performance Metrics:

Fast computation for small to moderate values of n (< 10,000)

Minimal memory footprint (typically kilobytes range)

Linear execution time for reasonable input sizes

Graph Features:

X-axis: values of n (1 to 50)

Y-axis: calculated λ(n) values

Markers at each data point for clarity

Grid lines for easier reading

Legend and title for context

DIFFICULTIES FACED BY THE STUDENT

Grasping the Carmichael Function Idea: uncertainty, about the reason the formula varies between powers of 2 and odd primes and the purpose of using LCM to merge the values.

Prime Factorization Logic: Implementing the trial division method correctly, especially handling the remaining factor after the loop for factors greater than √n.

LCM Computation: Ensuring the iterative LCM calculation correctly combines multiple values without errors in the accumulation.

Memory Management: Understanding and using the psutil library to accurately track process memory; confusion between RSS (resident set size) and other memory metrics.

Matplotlib Visualization:

Setting up the plot with appropriate figure size and styling

Understanding how to use markers, legends, and grid options

Saving the figure with the correct file format

Code Efficiency: Understanding that when n is large the prime factorization loop may become sluggish; exploring ways to optimize it.

Variable Naming: Employing abbreviated names (e.g. f v, p, k) complicated debugging; finding a compromise, between conciseness and readability.

Input Validation: Ensuring the program handles edge cases (n = 1, negative inputs) gracefully

SKILLS ACHIEVED

? Mathematical Programming: - Implementing number theory algorithms from mathematical definitions - Applying prime factorization techniques efficiently - Using GCD and LCM for composite calculations

? Algorithm Design: - Breaking down complex problems into modular functions - Implementing iterative loops for factorization - Combining multiple results using mathematical operations

? Python Proficiency: - Writing functions with clear logic flow - Using conditional statements for special cases - Iterating through data structures (lists, tuples) - String formatting for output display

? Performance Analysis: - Using time module for benchmarking - Integrating psutil for system resource monitoring - Understanding memory usage patterns

? Data Visualization: - Creating line plots with matplotlib - Adding labels, titles, legends, and grids - Saving figures to disk - Choosing appropriate visualization styles

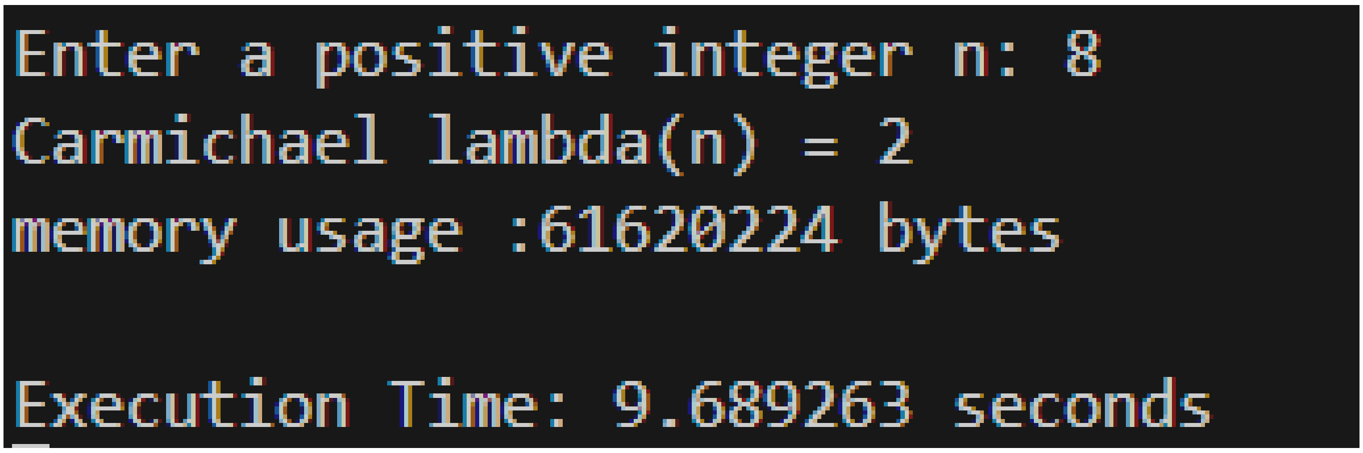
? Debugging & Testing: - Validating outputs against mathematical properties - Testing edge cases and special inputs - Tracing through algorithm execution

? Code Organization: - Modularizing code into well-defined functions - Reducing code duplication through helper functions - Maintaining logical flow from input to output

? Technical Communication: - Documenting code behavior and results - Understanding performance metrics - Creating meaningful visual representations

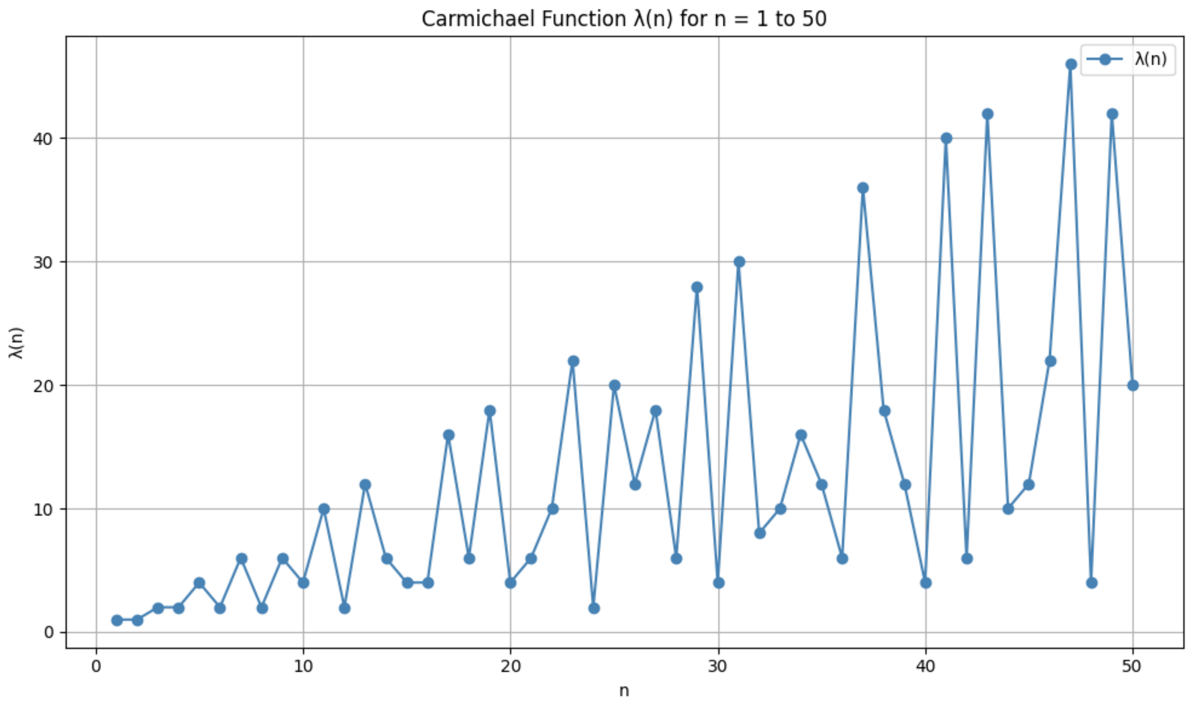
CONCLUSION

This project focused on the Carmichael function showcases the blend of number theory with real-world computational application. By converting mathematical ideas into functional Python programs the student acquired practical experience in algorithm development, efficiency tracking and visual data representation. The difficulties encountered—from grasping foundations to creating optimized algorithms—enhanced crucial abilities in mathematical coding and software engineering. Incorporating performance measurements alongside displays offers meaningful understanding of the behavior of abstract mathematical functions, over various inputs.



```
Enter a positive integer n: 8
Carmichael lambda(n) = 2
memory usage :61620224 bytes

Execution Time: 9.689263 seconds
```


Carmichael Function λ(n) for n = 1 to 50
λ(n)
λ(n)
n