Name: Devanshi Jain
Er-No:22162101006
Batch: 51
Branch: CBA

Institute of Computer Technology
B. Tech Computer Science and Engineering

Sub: Algorithm Analysis and Design

Practical 2

(1) MPSoft Technologies Pvt. Ltd. is a fast growing IT industry and wants to implement a function to calculate the monthly income generated from all projects from their N no of clients like C1,C2,C3,C4….CN. The team wants to compare the time/steps required to execute this function on various inputs and analyse the complexity of each combination. Also draw a comparative chart. In each of the following functions N will be passed by user.
Design the algorithm for the same and implement using the programming language of your choice. Make comparative analysis for various use cases & input size.

1. To calculate the sum of 1 to N number using loop.
2. To calculate the sum of 1 to N number using the equation.
3. To calculate sum of 1 to N numbers using recursion

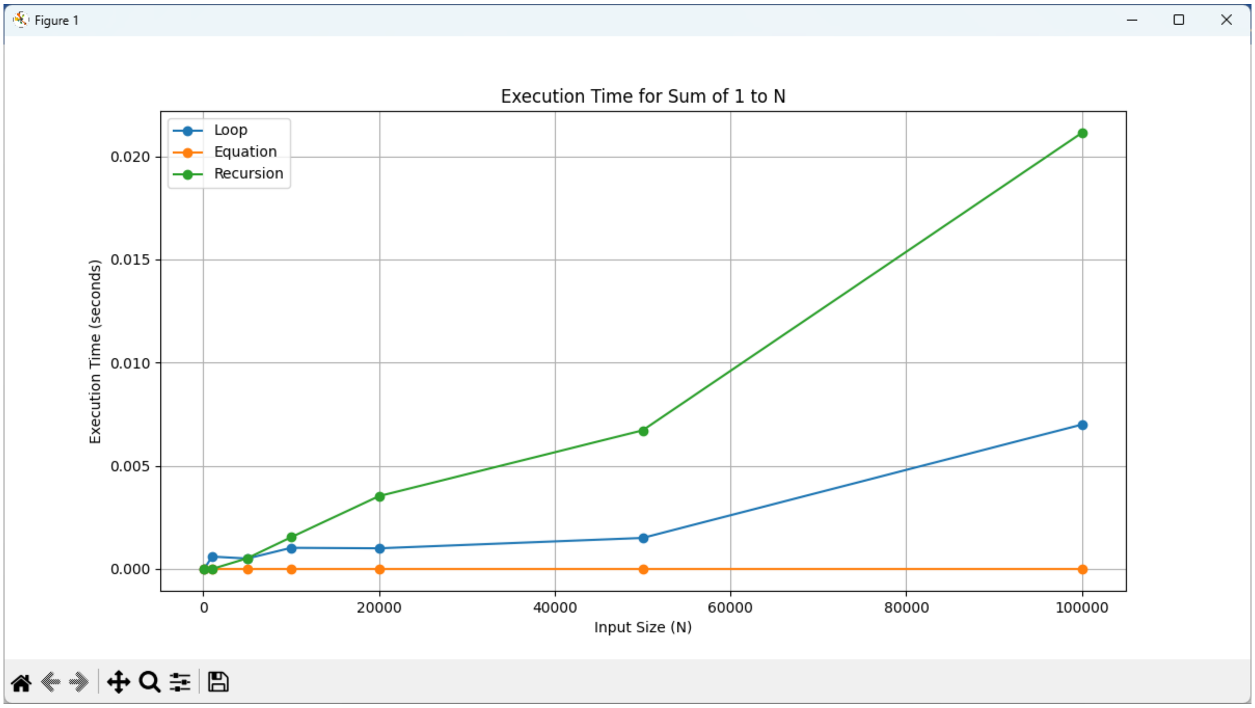
CODE:

```
import time
import matplotlib.pyplot as plt
import sys

# Increase the recursion limit for large inputs
sys.setrecursionlimit(1000000)

# Function to calculate sum using a loop
def sum_using_loop(N):
    total = 0
    for i in range(1, N + 1):
```

```
        total += i
    return total

# Function to calculate sum using the formula
def sum_using_equation(N):
    return N * (N + 1) // 2

# Function to calculate sum using recursion
def sum_using_recursion(N):
    if N == 1:
        return 1
    return N + sum_using_recursion(N - 1)

# Function to measure execution time
def measure_time(func, N):
    start_time = time.time()
    try:
        func(N)
    except RecursionError:
        return float('inf')
    end_time = time.time()
    return end_time - start_time

# Define the input sizes for testing
input_sizes = [100, 1000, 5000, 10000, 20000, 50000, 100000]
loop_times = []
equation_times = []
recursion_times = []

# Measure execution times for each method and input size
for size in input_sizes:
    loop_times.append(measure_time(sum_using_loop, size))
    equation_times.append(measure_time(sum_using_equation, size))
    recursion_times.append(measure_time(sum_using_recursion, size))

# Plot the results
plt.figure(figsize=(12, 6))
plt.plot(input_sizes, loop_times, label='Loop', marker='o')
plt.plot(input_sizes, equation_times, label='Equation', marker='o')
plt.plot(input_sizes, recursion_times, label='Recursion', marker='o')

plt.xlabel('Input Size (N)')
plt.ylabel('Execution Time (seconds)')
plt.title('Execution Time for Sum of 1 to N')
plt.legend()
```

```
plt.grid(True)
plt.show()
```

OUTPUT:



(2) Suppose a newly-born pair of rabbits, one male, one female, are put in a field. Rabbits are able to mate at the age of one month so that at the end of its second month a female can produce another pair of rabbits. Suppose that our rabbits never die and that the female always produces one new pair (one male, one female) every month from the second month on. How many pairs will there be in one year? Apply appropriate algorithm/method to find out the above problem and also solve them using iteration and recursive method. Compare the performance of two methods by counting the number of steps executed on various inputs. Also draw a comparative chart.

Design the algorithm for the same and implement using the programming language of your choice. Make comparative analysis for various use cases & input size.
CODE:

```
import time
import matplotlib.pyplot as plt

# Function to calculate Fibonacci number iteratively
def fibonacci_iterative(n):
    if n <= 1:
        return n
    a, b = 0, 1
    for _ in range(2, n + 1):
        a, b = b, a + b
    return b

# Function to calculate Fibonacci number recursively
def fibonacci_recursive(n):
    if n <= 1:
        return n
    return fibonacci_recursive(n - 1) + fibonacci_recursive(n - 2)

# Function to measure execution time
def measure_time(func, n):
    start_time = time.time()
    func(n)
    end_time = time.time()
    return end_time - start_time

# Define the input sizes for testing
input_sizes = [5, 10, 15, 20, 25, 30, 35]
iterative_times = []
recursive_times = []

# Measure execution times for each method and input size
for size in input_sizes:
    iterative_times.append(measure_time(fibonacci_iterative, size))
    recursive_times.append(measure_time(fibonacci_recursive, size))
```

```python
# Calculate Fibonacci numbers for 12 months
n_months = 12
rabbit_pairs_iterative = fibonacci_iterative(n_months)
rabbit_pairs_recursive = fibonacci_recursive(n_months)

# Print the results
print(f"Number of rabbit pairs after {n_months} months (Iterative):
{rabbit_pairs_iterative}")
print(f"Number of rabbit pairs after {n_months} months (Recursive):
{rabbit_pairs_recursive}")

# Plot the results
plt.figure(figsize=(12, 6))
plt.plot(input_sizes, iterative_times, label='Iterative', marker='o')
plt.plot(input_sizes, recursive_times, label='Recursive', marker='o')

plt.xlabel('Input Size (n)')
plt.ylabel('Execution Time (seconds)')
plt.title('Execution Time for Fibonacci Calculation')
plt.legend()
plt.grid(True)

plt.show()
```

OUTPUT: