Name : Harpalsinh Bhati

Enr. No.: 22162171010

Sub: Algorithm Analysis & Design

Branch: CS

Batch: 54

Sem: 5-B

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

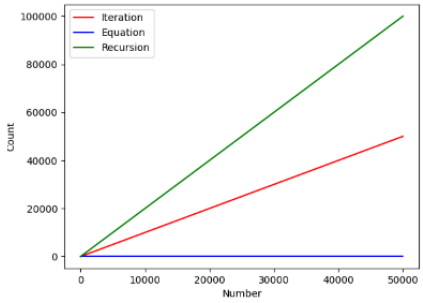
Code:

```
from flask import Flask, render_template
import matplotlib.pyplot as plt
import io
import base64
import sys

app = Flask(__name__)

sys.setrecursionlimit(100000)

@app.route('/')
def index():
    lst = [10, 100, 1000, 5000, 10000, 50000]
    sum1 = 0
    sum2 = 0
    sum3 = 0
    count_for_loop = []
    count_for_equation = []
    count_for_recursion = []
```

```
    for i in lst:
        loop = 0
        for j in range(1, i + 1):
            sum1 += 1
            loop += 1
        count_for_loop.append(loop)

        equation = 1
        sum2 = (i * (i + 1)) / 2
        count_for_equation.append(equation)

        recursion = [0]
        sum3 = sumUsingRecursion(i, recursion)
        count_for_recursion.append(recursion[0])

    # Plotting
    plt.figure()
    plt.plot(lst, count_for_loop, color='r', linestyle='-', label='Iteration')
    plt.plot(lst, count_for_equation, color='b', linestyle='-',
label='Equation')
    plt.plot(lst, count_for_recursion, color='g', linestyle='-',
label='Recursion')
    plt.xlabel('Number')
    plt.ylabel('Count')
    plt.legend()

    img = io.BytesIO()
    plt.savefig(img, format='png')
    img.seek(0)
    plot_url = base64.b64encode(img.getvalue()).decode()

    return render_template('Prac_2_1.html', plot_url=plot_url,
                           numbers=lst,
                           count_loops=count_for_loop,
                           count_equations=count_for_equation,
                           count_recursions=count_for_recursion)

def sumUsingRecursion(n, counter):
    counter[0] += 1
    if n == 0:
        counter[0] += 1
        return 0
    else:
        counter[0] += 1
        return n + sumUsingRecursion(n - 1, counter)

if __name__ == '__main__':
    app.run(debug=True)
```

Output:

Sum Comparison Result



Counts Table

Number	Count (Iteration)	Count (Equation)	Count (Recursion)
10	10	1	22
100	100	1	202
1000	1000	1	2002
5000	5000	1	10002
10000	10000	1	20002
50000	50000	1	100002

(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:

```
from flask import Flask, render_template
import matplotlib.pyplot as plt
import io
import base64
import sys

app = Flask(__name__)

sys.setrecursionlimit(5000)

# Iterative Fibonacci
def fib_iterative(n):
    a, b = 0, 1
```

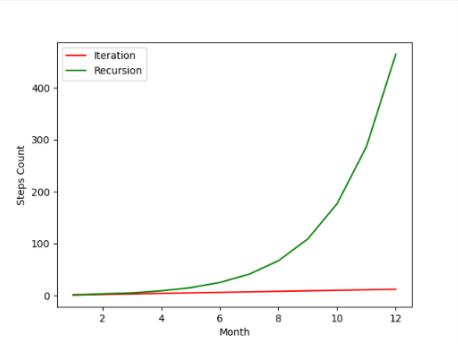
```
    steps = 0
    for _ in range(n):
        a, b = b, a + b
        steps += 1
    return a, steps

# Recursive Fibonacci
def fib_recursive(n, steps=0):
    steps += 1
    if n <= 1:
        return n, steps
    else:
        a, steps = fib_recursive(n - 1, steps)
        b, steps = fib_recursive(n - 2, steps)
        return a + b, steps

@app.route('/')
def index():
    months = range(1, 18)
    iterative_counts = []
    recursive_counts = []
    rabbit_pairs = []

    for month in months:
        rabbits_iter, steps_iter = fib_iterative(month)
        rabbits_rec, steps_rec = fib_recursive(month)

        rabbit_pairs.append(rabbits_iter)
        iterative_counts.append(steps_iter)
        recursive_counts.append(steps_rec)

    # Plotting the comparison
    plt.figure()
    plt.plot(months, iterative_counts, color='r', linestyle='-',
label='Iteration')
    plt.plot(months, recursive_counts, color='g', linestyle='-',
label='Recursion')
    plt.xlabel('Month')
    plt.ylabel('Steps Count')
    plt.legend()

    img = io.BytesIO()
    plt.savefig(img, format='png')
    img.seek(0)
    plot_url = base64.b64encode(img.getvalue()).decode()

    return render_template('Prac_2_2.html', plot_url=plot_url,
                           months=list(months),
```

```
                           rabbit_pairs=rabbit_pairs,
                           iterative_steps=iterative_counts,
                           recursive_steps=recursive_counts)

if __name__ == '__main__':
    app.run(debug=True)
```

Output:

Rabbit Population Growth

Comparative Analysis of Iterative vs Recursive Methods



Data Table

Month	Rabbit Pairs	Iterative Steps	Recursive Steps
1	1	1	1
2	1	2	3
3	2	3	5
4	3	4	9
5	5	5	15
6	8	6	25
7	13	7	41
8	21	8	67
9	34	9	109
10	55	10	177
11	89	11	287
12	144	12	465