**CS 201**

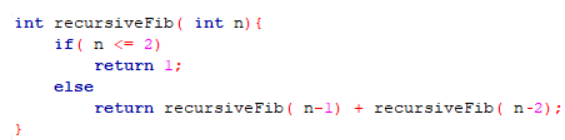
**HOMEWORK 2**

**Name: Muhammad Arham Khan**

**Section: 3**

**Bilkent ID: 21701848**

**THEORITICAL ANALYSIS**

* **Recursive algorithm:**

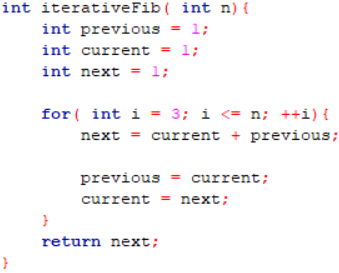
The algorithm involves uses the principle of recursion to output the Fibonacci number at location n. Considering time taken T(n) and T=2 units when n=1 or n=0 because returning is 1 unit and if comparison is 1 unit. So, the time taken for T(n) should be able to T(n-1) and T(n-2) due to the recursion calls to them in line 5. But considering the process time of the if condition and the return statement to be 1 unit each, the time taken is

**T(n) = T(n-1) + T(n-2) + 2**

Considering the formula and solving for various values of n like 2 (T = 4 = 22) or 3 (T= 8 = 23) gives an **exponential** trend that the Time complexity of the algorithm is of order 2n and hence:

**T(n) = O (2n)**

* **Iterative algorithm:**

This algorithm involves use of basic iteration to calculate the Fibonacci number of a number at location n. Considering the time taken to be T(n) and T = 4 units when n = 1 and 2 because the loop does not enter at these values of n and lines 2, 3, 4, 12 have constant time 1 unit.

Considering that inside loop, line 7 takes T = 2 units (one addition, one assignment), line 9 takes T = 1 unit (assignment) and line 10 takes T = 1 unit (assignment). Hence, the statements in for loop take constant time of T = 4 units regardless of n. But considering the iterations of loop, T = 4(n – 2) for the for loop and hence, the time taken for whole algorithm becomes:

**T(n) = 4(n-2) + 4**

And hence, considering the order of its time complexity, the algorithm produces a **linear** relation and hence,

**T(n) = O (n)**

**DATA COMPARISON**

**Recursive solution**

|  |  |  |  |
| --- | --- | --- | --- |
| **S#** | **Input value (n)** | **Theoretical value (2n)** | **Simulation value (milliseconds)** |
| 1 | 1 | 2 | 0.00001 |
| 2 | 5 | 32 | 0.000022 |
| 3 | 10 | 1024 | 0.000232 |
| 4 | 15 | 32768 | 0.003 |
| 5 | 20 | 1048576 | 0.036 |
| 6 | 25 | 33554432 | 0.387 |
| 7 | 30 | 1.07E+09 | 4.368 |
| 8 | 35 | 3.44E+10 | 48.484 |
| 9 | 40 | 1.1E+12 | 529 |
| 10 | 45 | 3.52E+13 | 5917 |
| 11 | 50 | 1.13E+15 | 66098 |
| 12 | 55 | 3.6E+16 | 764550 |

**Iterative solution**

|  |  |  |  |
| --- | --- | --- | --- |
| **S#** | **Input value (n)** | **Theoretical value (n)** | **Simulation value (milliseconds)** |
| 1 | 1 | 1 | 0.000003 |
| 2 | 50000000 | 50000000 | 180 |
| 3 | 100000000 | 100000000 | 361 |
| 4 | 150000000 | 150000000 | 567 |
| 5 | 200000000 | 200000000 | 743 |
| 6 | 250000000 | 250000000 | 917 |
| 7 | 300000000 | 300000000 | 1146 |
| 8 | 350000000 | 350000000 | 1295 |
| 9 | 400000000 | 400000000 | 1443 |
| 10 | 450000000 | 450000000 | 1637 |
| 11 | 500000000 | 500000000 | 1830 |
| 12 | 550000000 | 550000000 | 2071 |
| 13 | 600000000 | 600000000 | 2251 |
| 14 | 650000000 | 650000000 | 2493 |
| 15 | 700000000 | 700000000 | 2838 |
| 16 | 750000000 | 750000000 | 2965 |
| 17 | 800000000 | 800000000 | 2968 |
| 18 | 850000000 | 850000000 | 3209 |
| 19 | 900000000 | 900000000 | 3475 |
| 20 | 950000000 | 950000000 | 3495 |
| 21 | 1000000000 | 1000000000 | 3776 |

**PLOT COMPARISON**

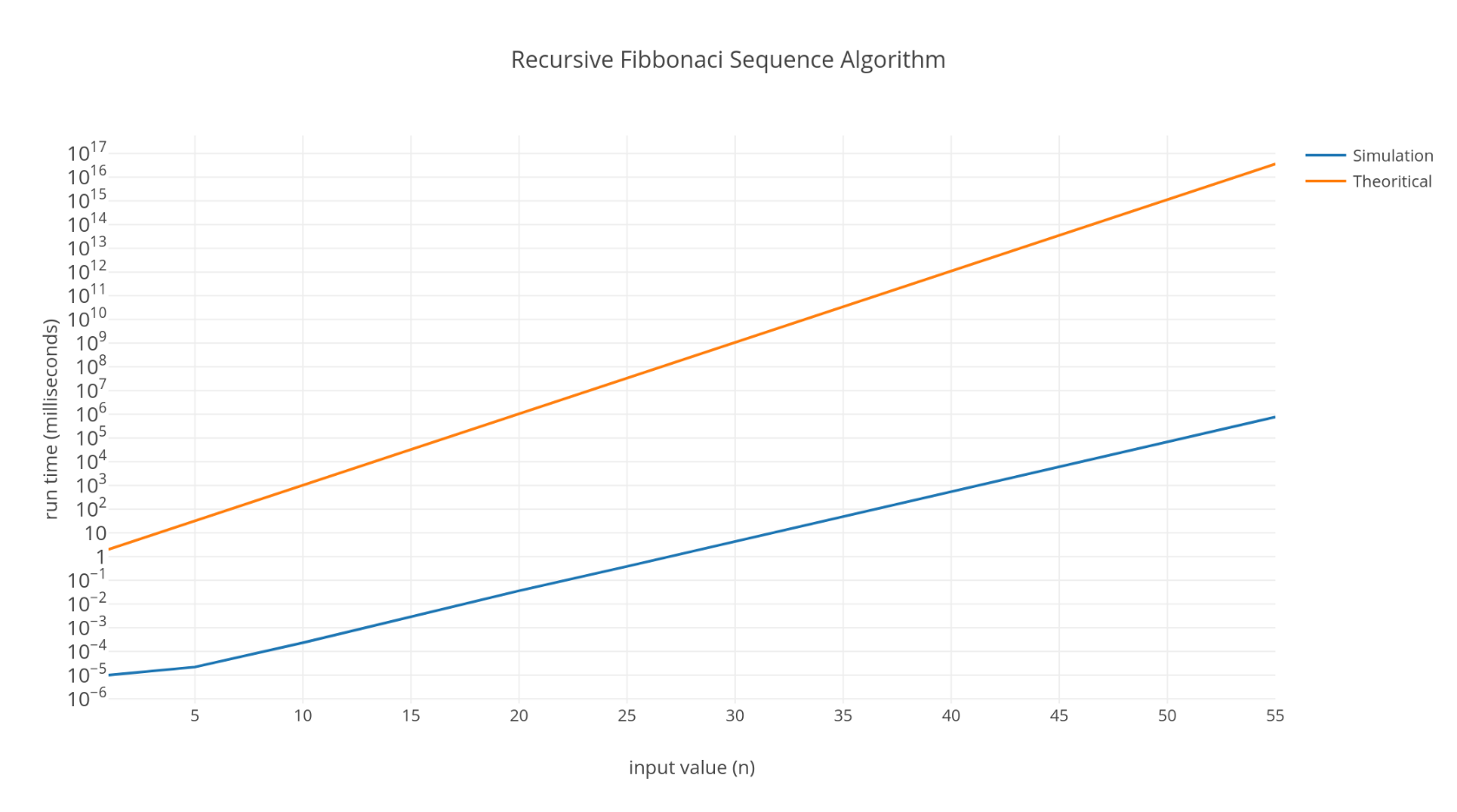
****

Figure 1: Recursive algorithm

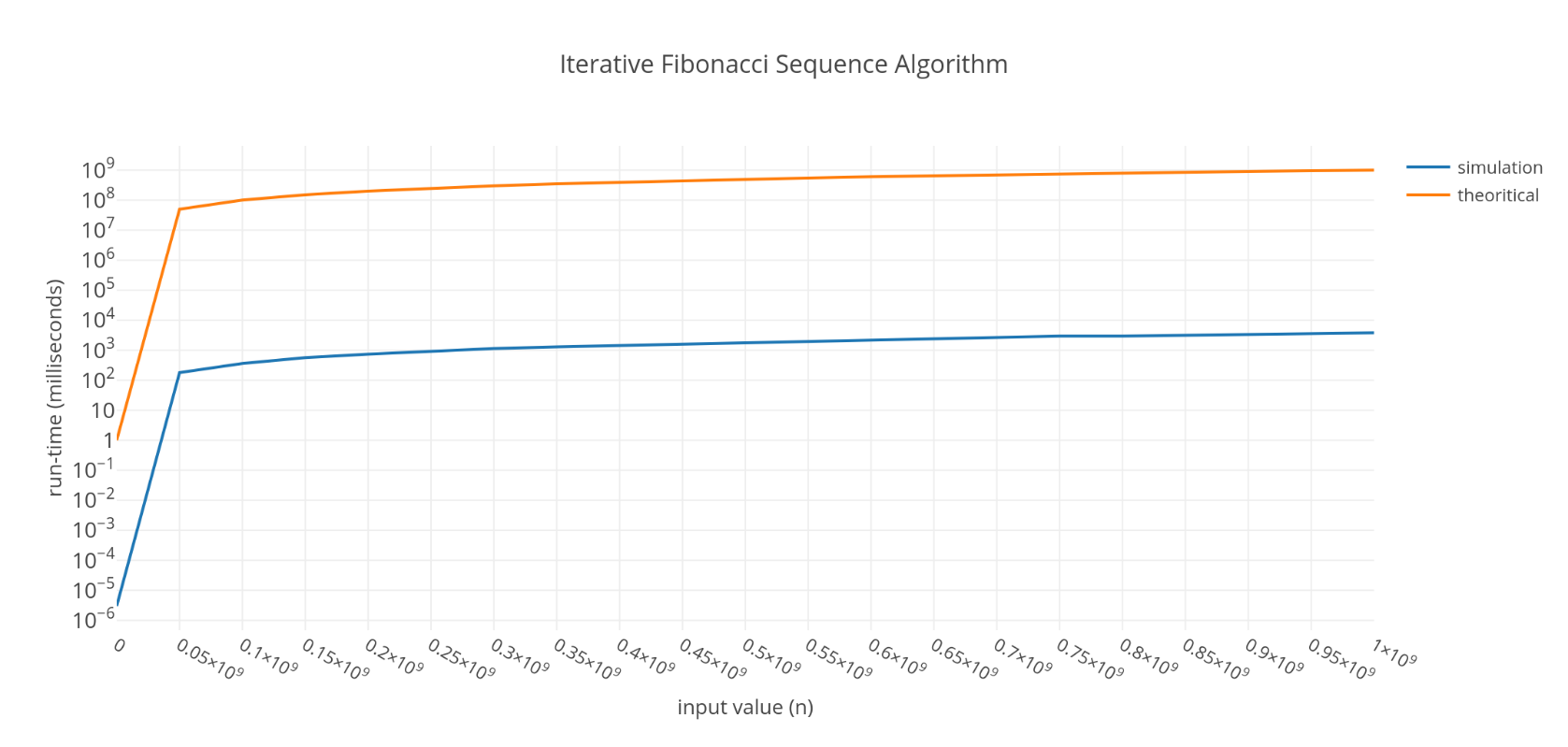
****

Figure 2: Iterative algorithm

**Note: the y-axis has logarithmic scale in both graphs**

**OBSERVATIONS**

* **Recursion algorithm (simulation vs. expected):**

As shown from the plots, this algorithm’s simulation follows the same exponential trend against values of n as the expected values (2n) and the time taken grows exponentially as n increases. But one thing is evident that the simulation times are lower as compared to the expected times for all values of n.

So, on a logarithmic y-axis plot, the algorithms exponential run time produces a linear growth as shown in Figure 1.

* **Iterative algorithm (simulation vs. expected):**

As shown from the plots, this algorithm’s simulation follows the same linear trend against values of n as the expected values (n) and the time taken grows linearly as the value of n increases. But one thing is evident that the simulation times are lower as compared to the expected times for all values of n.

So, on a logarithmic y-axis plot, the algorithms linear run time produces a bent graph as shown in Figure 2.

* **Comparing both algorithms:**

The iterative algorithm is much more efficient than the recursive one because the iterative algorithm’s run-times increase linearly (n = 1,000,000,000 taking 3776 milliseconds only) while recursive algorithm’s run-times increase exponentially (n = 55 taking over 764550 milliseconds) due to extra redundant processing and function calls in the recursive algorithm.

**COMPUTER SPECIFICATIONS**

* Core i7 (7th gen) ~ 2.2GHz
* 16GB DDR4 RAM
* Nvidia 940MX 2GB graphics
* OS: Microsoft Windows 10 - Education