# Cs 201 Homework 2

## Recursive Calls

### Recursive Fibonacci Sequence Between 5 - 55 ( Figure 1 ) - Simulation 1

```
Recursive of f(5) : 5
Execution took 0.045 milliseconds.
Program ended with exit code: 0

Recursive of f(10) : 55
Execution took 0.053 milliseconds.
Program ended with exit code: 0

Recursive of f(15) : 610
Execution took 0.046 milliseconds.
Program ended with exit code: 0

Recursive of f(20) : 6765
Execution took 0.081 milliseconds.
Program ended with exit code: 0

Recursive of f(25) : 75025
Execution took 0.459 milliseconds.
Program ended with exit code: 0

Recursive of f(30) : 832040
Execution took 4.686 milliseconds.
Program ended with exit code: 0

Recursive of f(35) : 9227465
Execution took 45.562 milliseconds.
Program ended with exit code: 0

Recursive of f(40) : 102334155
Execution took 491.417 milliseconds.
Program ended with exit code: 0

Recursive of f(45) : 1134903170
Execution took 5070.16 milliseconds.
Program ended with exit code: 0

Recursive of f(50) : 12586269025
Execution took 55099.1 milliseconds.
Program ended with exit code: 0

Recursive of f(55) : 139583862445
Execution took 659971 milliseconds.
Program ended with exit code: 0
```

## Table 1 of Figure 1

| Input Size (N) | Running Time |
|---|---|
| **5** | 0.045 milliseconds |
| **10** | 0.053 milliseconds |
| **15** | 0.046 milliseconds |
| **20** | 0.081 milliseconds |
| **25** | 0.459 milliseconds |
| **30** | 4.6886 milliseconds |
| **35** | 45.562 milliseconds |
| **40** | 491.417 milliseconds |
| **45** | 5070.16 milliseconds |
| **50** | 55099.1 milliseconds |
| **55** | 659971 milliseconds |

# Iterative Calls

## Iterative Fibonacci Sequence Between 5 - 55 ( Figure 2 ) - Simulation 2

```
Iterative of f(5) : 5
Execution took 0.045 milliseconds.
Program ended with exit code: 0

Iterative of f(10) : 55
Execution took 0.043 milliseconds.
Program ended with exit code: 0

Iterative of f(15) : 610
Execution took 0.046 milliseconds.
Program ended with exit code: 0

Iterative of f(20) : 6765
Execution took 0.046 milliseconds.
Program ended with exit code: 0

Iterative of f(25) : 75025
Execution took 0.043 milliseconds.
Program ended with exit code: 0

Iterative of f(30) : 832040
Execution took 0.04 milliseconds.
Program ended with exit code: 0

Iterative of f(35) : 9227465
Execution took 0.043 milliseconds.
Program ended with exit code: 0

Iterative of f(40) : 102334155
Execution took 0.044 milliseconds.
Program ended with exit code: 0

Iterative of f(45) : 1134903170
Execution took 0.042 milliseconds.
Program ended with exit code: 0

Iterative of f(50) : 12586269025
Execution took 0.048 milliseconds.
Program ended with exit code: 0

Iterative of f(55) : 139583862445
Execution took 0.046 milliseconds.
Program ended with exit code: 0
```

## Table 2 of Figure 2

| Input Size (N) | Running Time |
|---|---|
| **5** | 0.043 milliseconds |
| **10** | 0.046 milliseconds |
| **15** | 0.046 milliseconds |
| **20** | 0.046 milliseconds |
| **25** | 0.043 milliseconds |
| **30** | 0.04 milliseconds |
| **35** | 0.043 milliseconds |
| **40** | 0.044 milliseconds |
| **45** | 0.042 milliseconds |
| **50** | 0.048 milliseconds |
| **55** | 0.046 milliseconds |

## Iterative Fibonacci Sequence Between 100 - 1.000.000.000 ( Figure 3 ) - Simulation 3

```
Iterative of f(100) : 3736710778780434371
Execution took 0.044 milliseconds.
Program ended with exit code: 0

Iterative of f(1000) : 817770325994397771
Execution took 0.043 milliseconds.
Program ended with exit code: 0

Iterative of f(10000) : -2872092127636481573
Execution took 0.084 milliseconds.
Program ended with exit code: 0

Iterative of f(100000) : 2754320626097736315
Execution took 0.369 milliseconds.
Program ended with exit code: 0

Iterative of f(1000000) : -4249520595888827205
Execution took 3.501 milliseconds.
Program ended with exit code: 0

Iterative of f(10000000) : -8398834052292539589
Execution took 31.137 milliseconds.
Program ended with exit code: 0

Iterative of f(100000000) : -4307732722963583941
Execution took 311.045 milliseconds.
Program ended with exit code: 0

Iterative of f(1000000000) : 3311503426941990459
Execution took 2894.86 milliseconds.
Program ended with exit code: 0
```

### Table 3 of the Figure 3

| Input Size (N) | Running Time |
|---|---|
| **100** | 0.044 milliseconds |
| **1000** | 0.043 milliseconds |
| **10000** | 0.084 milliseconds |
| **100000** | 0.369 milliseconds |
| **1000000** | 3.501 milliseconds |
| **10000000** | 31.137 milliseconds |
| **100000000** | 311.045 milliseconds |
| **1000000000** | 2894.86 milliseconds |

## Iterative Fibonacci Sequence Between 100.000.000 - 1.000.000.000 ( Figure 4 ) - Simulation 4

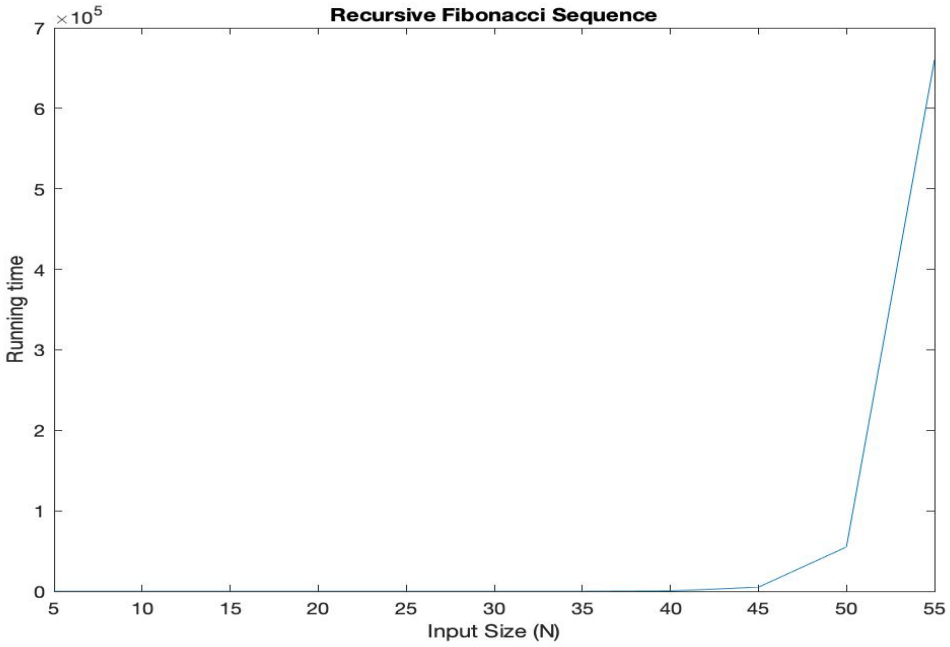
```
Iterative of f(100000000) : -4307732722963583941
Execution took 309.104 milliseconds.
Program ended with exit code: 0

Iterative of f(200000000) : 8108523128430920133
Execution took 619.71 milliseconds.
Program ended with exit code: 0

Iterative of f(300000000) : -2320241383415471104
Execution took 919.795 milliseconds.
Program ended with exit code: 0

Iterative of f(400000000) : -4584691783473964485
Execution took 1238.27 milliseconds.
Program ended with exit code: 0

Iterative of f(500000000) : 4055680951918129093
Execution took 1547.05 milliseconds.
Program ended with exit code: 0

Iterative of f(600000000) : 7227948606491740160
Execution took 1851.64 milliseconds.
Program ended with exit code: 0

Iterative of f(700000000) : 8880746853578164283
Execution took 2175.7 milliseconds.
Program ended with exit code: 0

Iterative of f(800000000) : 9062594501314731461
Execution took 2476.04 milliseconds.
Program ended with exit code: 0

Iterative of f(900000000) : -2694463113204044800
Execution took 2788.77 milliseconds.
Program ended with exit code: 0

Iterative of f(1000000000) : 3311503426941990459
Execution took 3077.32 milliseconds.
Program ended with exit code: 0
```

### Table 4 of Figure 4

| Input Size (N) | Running Time |
|---|---|
| **100000000** | 309.104 milliseconds |
| **200000000** | 619.71 milliseconds |
| **300000000** | 919.795 milliseconds |
| **400000000** | 1238.27 milliseconds |
| **500000000** | 1547.05 milliseconds |
| **600000000** | 1851.64 milliseconds |
| **700000000** | 2175.7 milliseconds |
| **800000000** | 2476.04 milliseconds |
| **900000000** | 2788.77 milliseconds |
| **1000000000** | 3077.32 milliseconds |

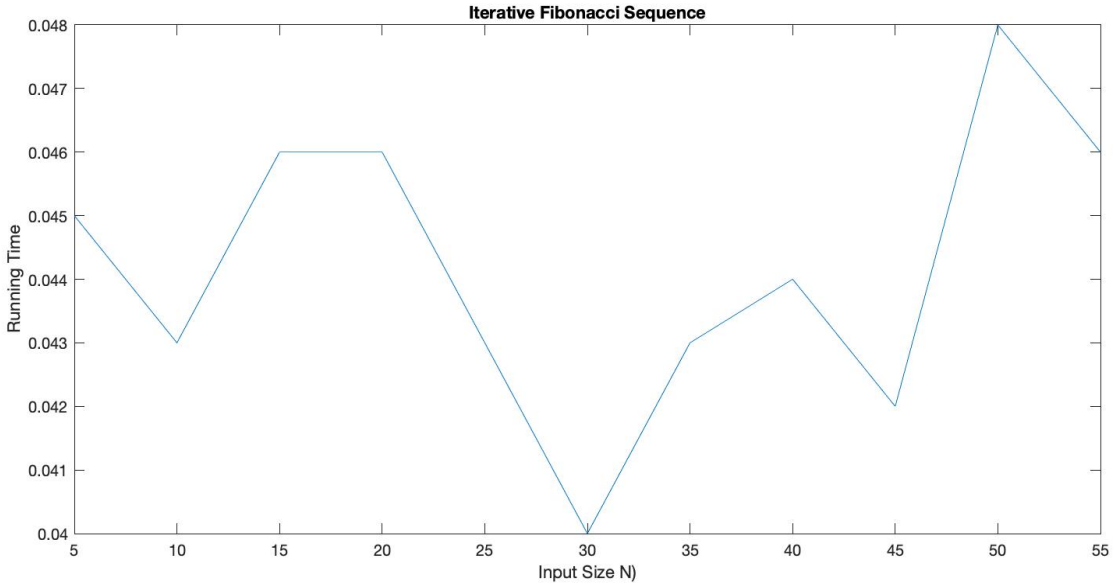
## Obtained Results

- **Figure 1**



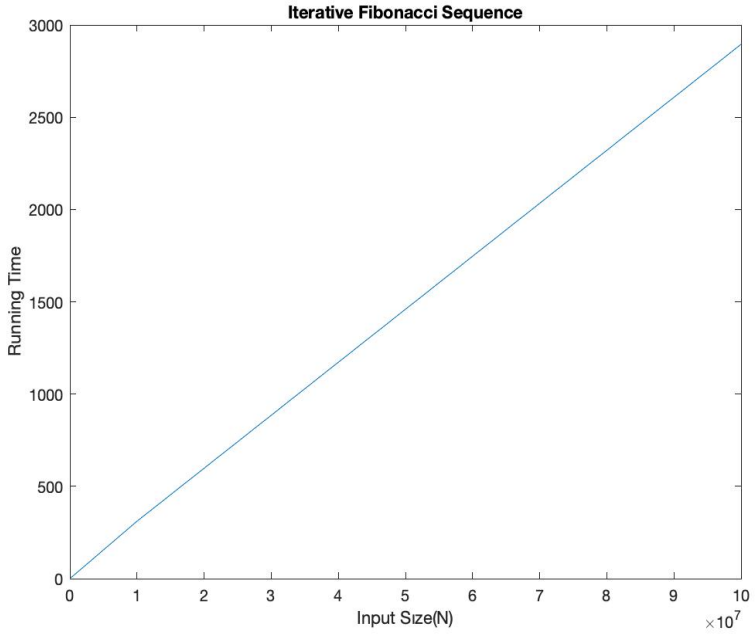

**This figure is the input size versus Running time of the Recursive Fibonacci Sequence algorithm which's input size is between 1 and 55.**

- **Figure 2**



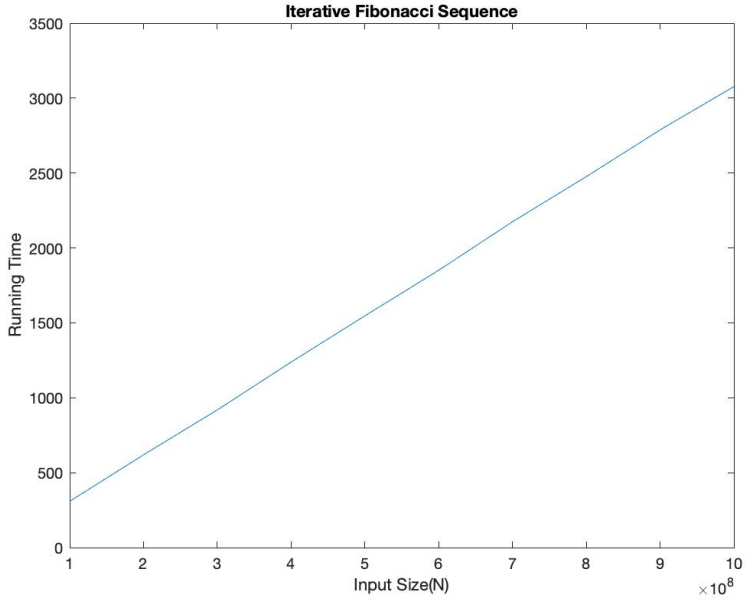

**This figure is the input size versus Running time of the Iterative Fibonacci Sequence algorithm which's input size is between 1 and 55.**

- **Figure 3**




**This figure is the input size versus Running time of the Iterative Fibonacci Sequence algorithm which's input size is between 1 and 1.000.000.000.**

- **Figure 4**




**This figure is the input size versus Running time of the Iterative Fibonacci Sequence algorithm which's input size is between 100.000.00 and 1.000.000.000.**

**Specifications of the computer that used for obtain execution times of the iterative and recursive Fibonacci Sequence algorithms**

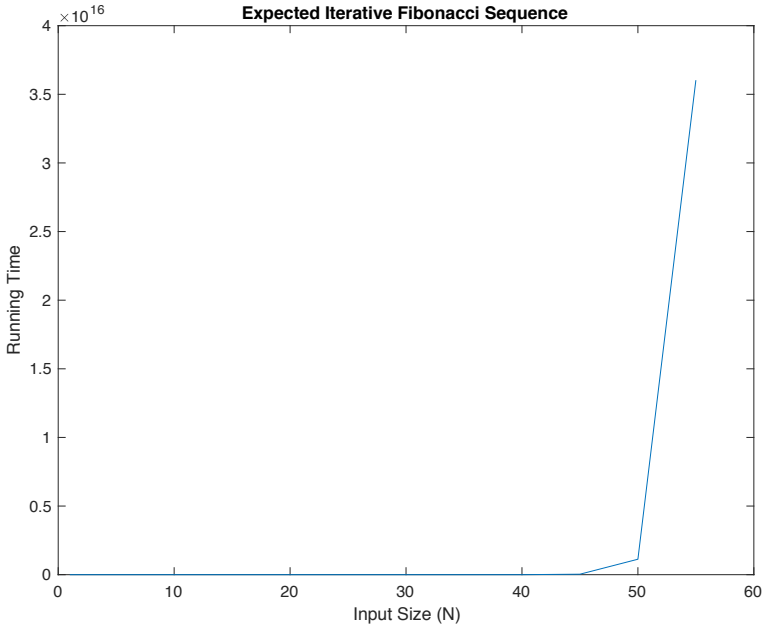
**Hardware Overview:**

| | |
|---|---|
| Model Name: | MacBook Pro |
| Model Identifier: | MacBookPro15 |
| Processor Name: | Intel Core i7 |
| Processor Speed: | 2,6 GHz |
| Number of Processors: | 1 |
| Total Number of Cores: | 4 |
| L2 Cache (per Core): | 256 KB |
| L3 Cache: | 6 MB |
| Memory: | 16 GB |
| Graphics: | Intel HD Graphics 530 1536 MB |
| Version: | macOS Mojave 10.14 |

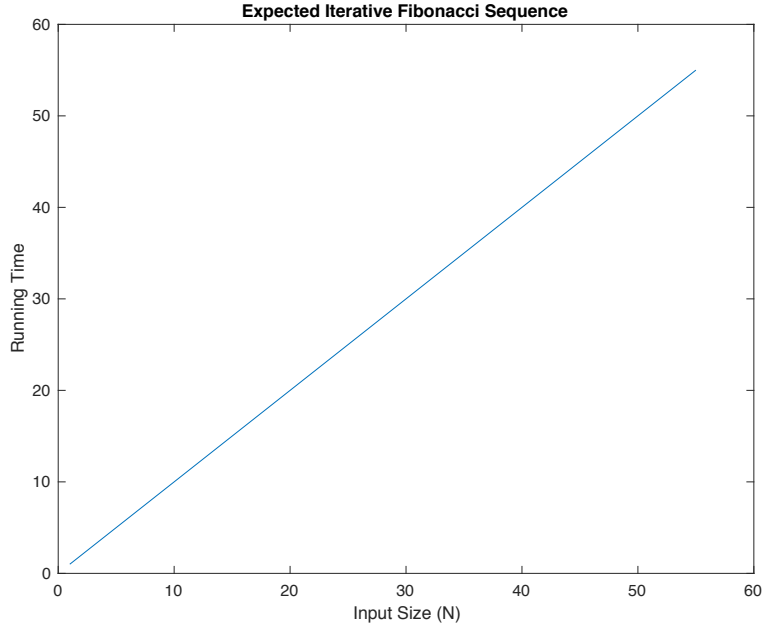
## Expected Growth Rates Obtained From the Theoretical Analysis

The expected upper bounds for the recursive running time is O ( 2 ^ N ), and the iterative running time is O ( N ). In the following lines, the expected running times of the simulations are plotted.

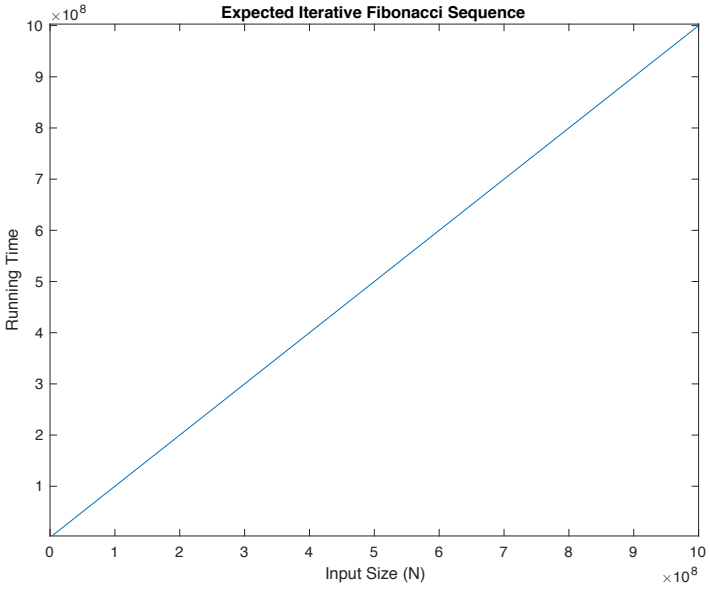
- **Expected Growth Rate of Simulation 1**



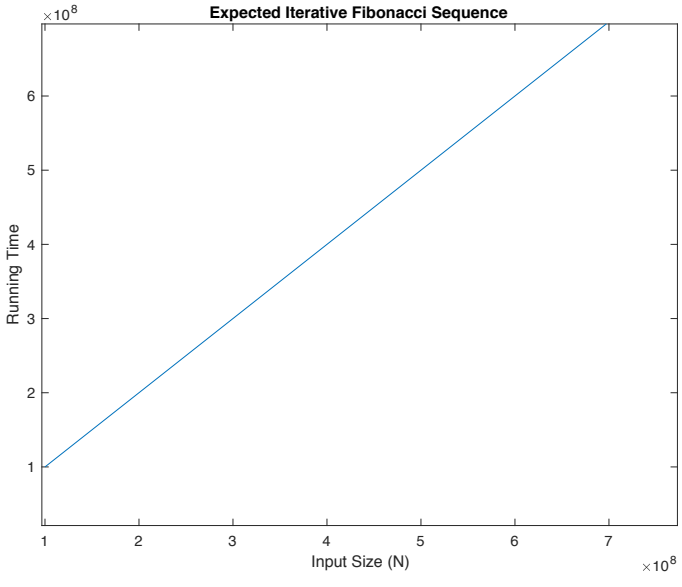

- **Expected Growth Rate of Simulation 2**



- **Expected Growth Rate of Simulation 3**




- **Expected Growth Rate of Simulation 4**



The expected growth rate of iterative Fibonacci Sequence is o(N), and it is linear. Since iterative method of Fibonacci Sequence is implemented by one for loop the running time of it is the running time of the statements İn the for loop which is N.

```
for(int i = 3 ; i <= number ; i++)
```

On the other hand, recursive implementation of Fibonacci sequence is exponential. Analyze of the recursion can be done by induction. The function calls itself two times

```
return recursive(number-1) + recursive(number-2) ;
```

Which equals to,

T(N) = T(N - 1) + T (N - 2) -> Recurrence relation

Since the recursion calls itself two times, it ind the time-complexity of the algorithm is O(2^N). So it grows exponentially. In Fibonacci Sequence algorithms, using iterative solution is more efficient than recursive implementation. In recursive implementation, Fib(55) takes approximately 659971 millisecond. But in iterative solution Fib(55) takes likely 0.046 milliseconds. Hence Fib(1000000000) takes 3077.22 milliseconds.This difference occurs because, iterative implementation takes O(N) time, however recursive implementation takes O(2^N) time. As a result, using an iterative algorithm is much more efficient and reduces running time than the recursive Fibonacci Sequence Algorithm.

The simulation 1 is about the recursive implementation of Fibonacci sequence between input size 1 and 55. After the input size 45, Running times increased rapidly as compared to the before. Because recursive algorithms running time is O(2^N).

The simulation 2 is about the iterative implementation of Fibonacci sequence between input size 1 and 55. Normally, it should be increased linearly. But, because of the CPU performance, memory which changes, and the iterative running time between 1, and 55 is really closer, the graphs does not seems like a linear graph. I did the simulation 2 to compare efficiency between the Iterative and Recursive implementations.

The simulation 3 is about the iterative implementation of Fibonacci sequence between input size 1 and 1.000.000.000. This graph is a linear graph of running time versus input size (N).

The simulation 4 is about the iterative implementation of Fibonacci sequence between input size 100.000.000. and 1.000.000.000. This graph is a linear graph of running time versus input size (N).

### The main.cpp

```
//
//  main.cpp
//  Homework2
//
//  Created by Burak Korkmaz on 3.12.2018.
//  Copyright © 2018 Burak Korkmaz. All rights reserved.
//

#include <iostream>
#include <ctime>

using namespace std;

long recursive(long number){
    if (number == 2 || number == 1  )
        return 1;
    else if(number == 0)
        return 0;
    else
        return recursive(number-1) + recursive(number-2) ;
}

long iterative(long number){
    long current = 1;
    long prev = 1;
    long result = 1;

    if(number == 0)
        result = 0;

    for(int i = 3 ; i <= number ; i++){
        result = current + prev;
        prev = current;
        current = result;
    }

    return result;
}

int main() {
```

```
    // Recursive

    //Store the starting time
    double duration;
    clock_t startTime = clock();
    //Code block

    cout <<"Recursive of f(" << 52 << ") : " << recursive(52) <<
endl; // For Recursive fibonacci sequence

//    cout <<"Iterative of f(" << 1000000000 << ") : "
<<iterative(1000000000) << endl;  // For Iterative fibonacci sequence

    //Compute the number of seconds that passed since the starting time
    duration = 1000 * double( clock() - startTime ) / CLOCKS_PER_SEC;
    cout << "Execution took " << duration << " milliseconds." << endl;


    return 0;
}
```