**CS201 – HW02**

**Olcay Akman**

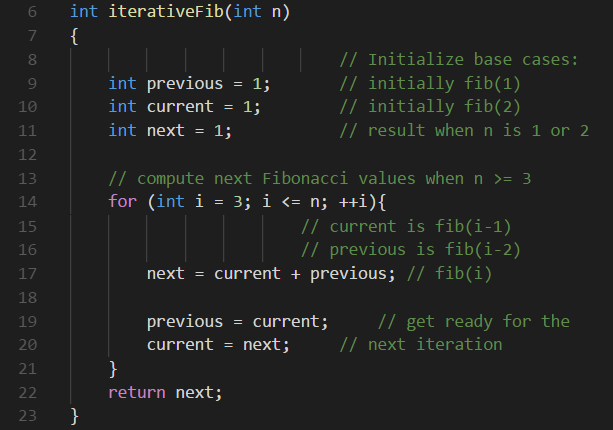
**ID: 21702671**

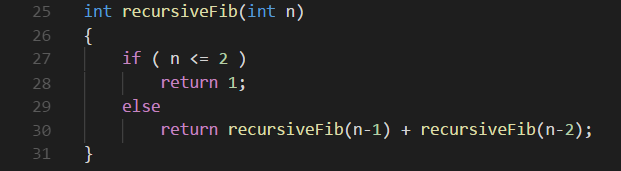
**Section 02**

**Introduction**

In this report, I will examine two alternative algorithms to the Fibonacci sequence and I will compare their time complexities in their implementations, along with the comparisons of their expected running times and the actual running times I have calculated using my computer.

The functions I have used to implement the two alternative solutions are given below.

 **Figure 1** – *The function used to implement the iterative algorithm for the Fibonacci sequence.*



**Figure 2** – *The function used to implement the recursive algorithm for the Fibonacci Sequence.*

The expected running time of the iterative function for the Fibonacci sequence is given by O(N) = log(N) and for the recursive function Fibonacci sequence O(2N) = log(2N). These functions can be used to estimate the expected running times of each function for randomly chosen N values.

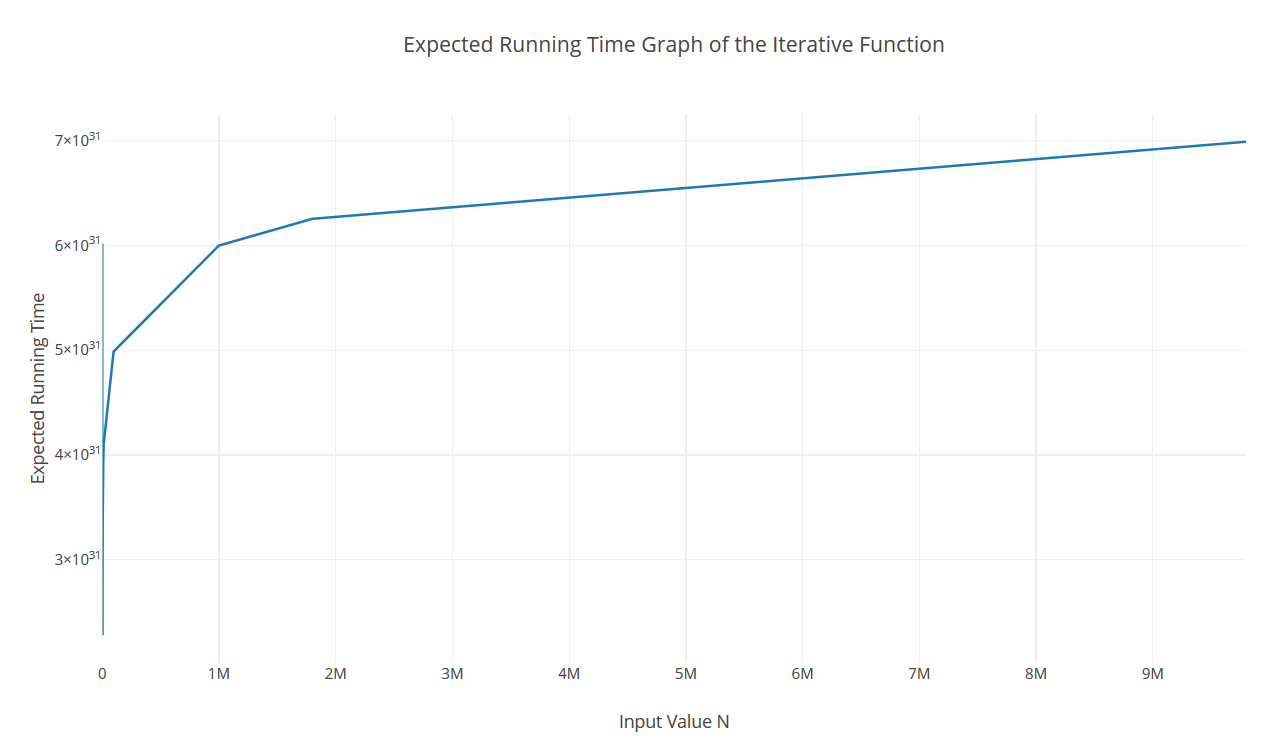
1. **The Iterative Function**

The following table shows the expected running time values for a few randomly chosen N values for the iterative function. (I have used a calculator to calculate the values given by the formulas given above, thus these are approximate values).

|  |  |
| --- | --- |
| N Value | Expected Running Time (milliseconds) |
| 40 | 1.602059991327962390427477789449 |
| 188 | 2.2741578492636798548416971888982 |
| 1007 | 3.0030294705536180071693257673842 |
| 8813 | 3.9451237701221194282659828271793 |
| 12774 | 4.1063269118219661988733778318958 |
| 97002 | 4.9867806886993314928971936119413 |
| 999001 | 5.9995659229549758410171831990663 |
| 1798337 | 6.2548710797922106413155399216456 |
| 9798337 | 6.9911523723241796247842942127699 |
| 99798337 | 7.9991233044362602524821265014996 |

**Figure 3** *– The table of the expected running times (milliseconds) for randomly chosen N values, given by the formula O(N) = log(N) for the iterative function for the Fibonacci Sequence.*

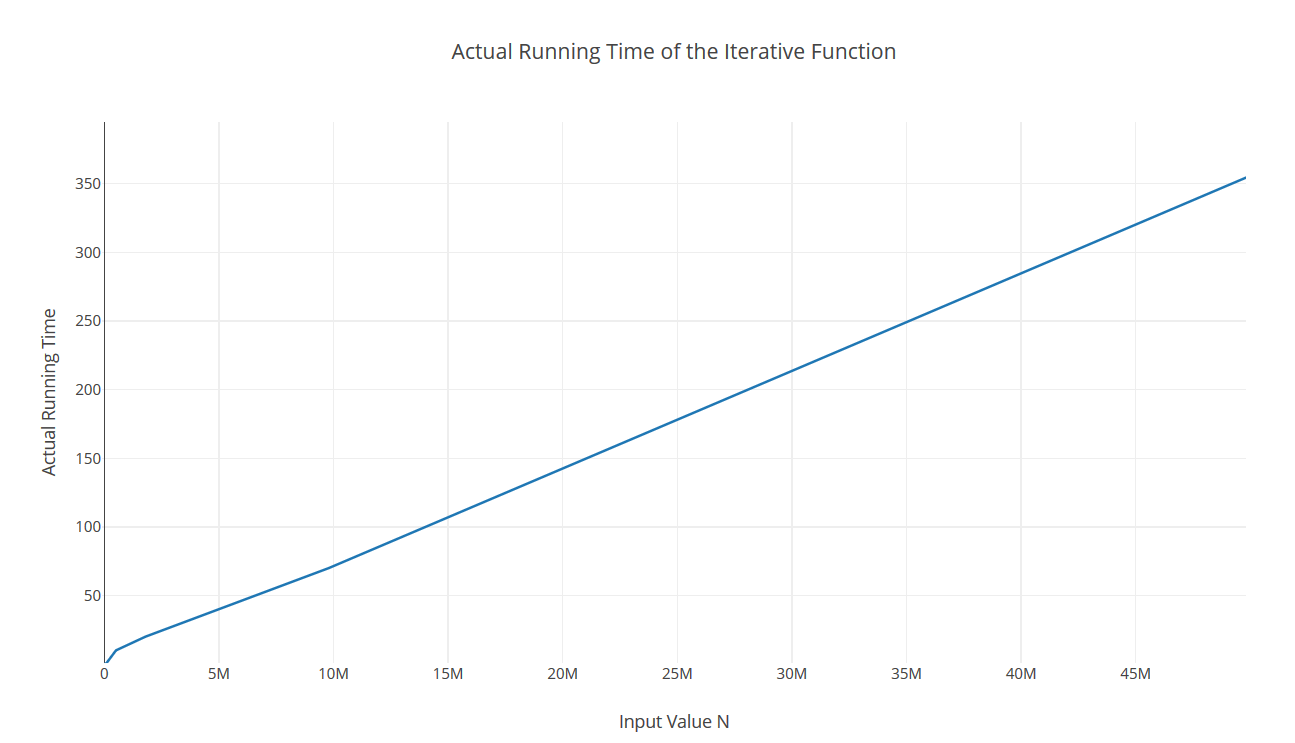
The following graph is for the table given above.

**Figure 4** – *The graph of the expected running times for randomly chosen N values, given by the formula O(N) = log(N) for the iterative function for the Fibonacci Sequence.*

The correctness of these values was tested by writing a main function to calculate the actual running time of the given function ran with the above-mentioned random values in Figure 1. The following table shows the actual running time values for the N values specified above.

|  |  |
| --- | --- |
| N Value | Actual Running Time |
| 40 | 0.000256 |
| 188 | 0.00126 |
| 1007 | 0.008 |
| 8813 | 0.06 |
| 12774 | 0.08 |
| 97002 | 0.72 |
| 999001 | 7.28 |
| 1798337 | 20 |
| 9798337 | 70 |
| 99798337 | 720 |

**Figure 5** – *The table of the actual running times for randomly chosen N values.*



The following graph is for the table given above.

**Figure 6** – *The graph of the actual running times for randomly chosen N values.*

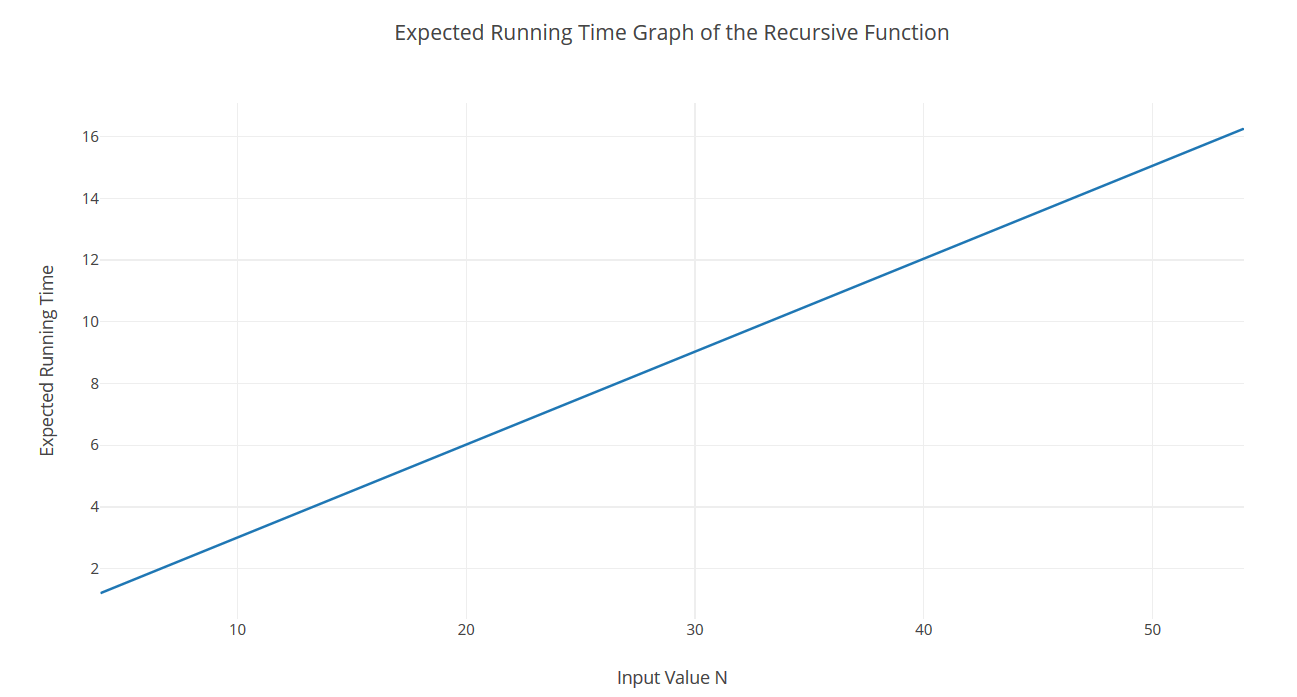
1. **The Recursive Function**

The following table shows the expected running time values for a few randomly chosen N values for the iterative function. (I have used a calculator to calculate the values given by the formulas given above, thus these are approximate values).

|  |  |
| --- | --- |
| N Value | Expected Running Time (milliseconds) |
| 4 | 1.204119982655924780854955578898 |
| 8 | 2.4082399653118495617099111577959 |
| 14 | 4.2144199392957367329923445261429 |
| 17 | 5.1175099262876803186335612103164 |
| 24 | 7.2247198959355486851297334733878 |
| 32 | 9.6329598612473982468396446311838 |
| 39 | 11.740169830895266613335816894255 |
| 44 | 13.245319809215172589404511367878 |
| 51 | 15.352529778863040955900683630949 |
| 54 | 16.255619765854984541541900315123 |

**Figure 7** *– The table of the expected running times (milliseconds) for randomly chosen N values, given by the formula O(N) = log(2N) for the recursice function for the Fibonacci Sequence.*

The following graph is for the table given above.



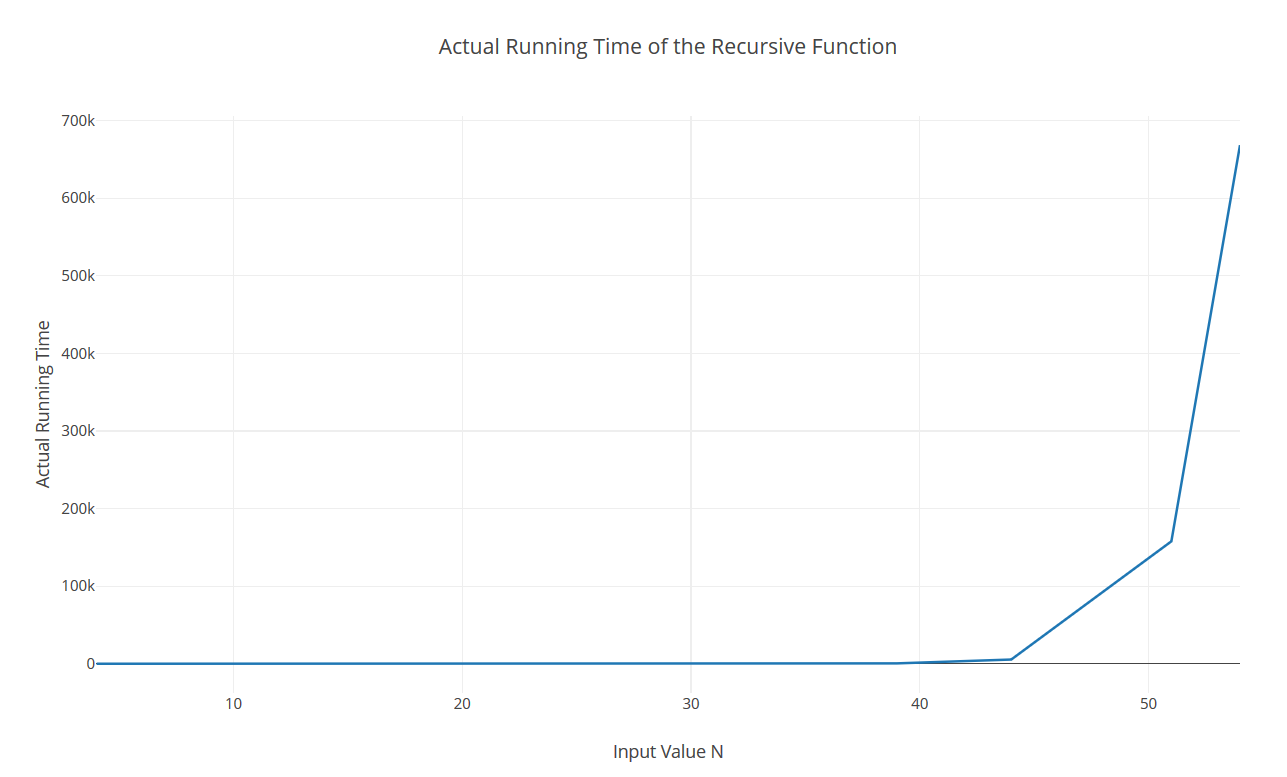
**Figure 8** – *The graph of the expected running times for randomly chosen N values, given by the formula O(N) = log(2N) for the recursice function for the Fibonacci Sequence.*

The correctness of these values was tested by writing a main function to calculate the actual running time of the given function ran with the above-mentioned random values in Figure 1. The following table shows the actual running time values for the N values specified above.

|  |  |
| --- | --- |
| N Value | Actual Running Time |
| 4 | 0.001 |
| 8 | 0.002 |
| 14 | 0.003 |
| 17 | 0.0124 |
| 24 | 0.358 |
| 32 | 20 |
| 39 | 490 |
| 44 | 5430 |
| 51 | 157780 |
| 54 | 668450 |

**Figure 9** – *The table of the actual running times for randomly chosen N values.*

The following graph is for the table given above.

**Figure 10** – *The graph of the actual running times for randomly chosen N values.*****

1. **Reflection**

In the first two sections of this report I have given the tables and the graphs for some randomly chosen N values to calculate the Fibonacci sequence. In order to do this, I have used two different algorithms. The first section of this report examined the iterative implementation function whereas the second section examined the recursive one.

In the iterative implementation, the increasing values of N as inputs were expected to give outputs that are increasing logarithmically. Although the graph we obtained by the actual running times is increasing, it does not behave *completely* like the expected graph, if we zoom in to the very beginning of the actual running time graph, given in Figure 6, for the smaller values of N, we will see that the graph does imitate the behavior in the expected running time graph given in Figure 4. The results do not match 100% accurately, but the values given Figure 5 do change with each execution, even if the computer is used.

The recursive implementation and its comparison to the expected outcomes yield similar results to what we see in the iterative implementation. The graph obtained by the implementation of the function, given in Figure 8, gives a result that seems rather different than the expected graph, given in Figure 10. However, we see that they both tend to increase with the increasing N values as their inputs. As mentioned for the iterative implementation, the results of the table in Figure 9 do change for each execution, as well. This is due to many factors including the features of the computer and the programming language used to implement these functions.

I have used the Dijkstra account provided to us at the beginning of this semester to execute my code, which I have written in the C++ language. These factors do affect the actual running time of either function I have used in my implementation. Further, the person who writes the code and how they turn these algorithms to their programming language equivalents also do affect the results, because the same algorithm may have different ways of implementations of different running times.

These two factors mainly affect the results we obtain from each execution of the function in Figure 1 and Figure 2. Even if these implementations were repeated in the same computer with the exact same code, we will likely get different results. (Which case I did) Another effect I will mention on the different results obtained by each execution is the values I have used to calculate them. I chose the N values randomly in this report. I could have chosen different values, more values or less values to do my implementations. This too, is an important factor to consider while comparing the expected and actual running time values.

A final factor that affects the running time is the method we use to calculate the runtime inside the driver(main) function while executing the functions. In my implementations, I used the Time.h file provided by the C++ Standard Library. I could have used another method to calculate the running time of these functions, this does affect the results, as well.

The running time of a function is different, depending on its character, such as iterative or recursive. Iterative functions’ runtimes increase logarithmically whereas recursive functions’ runtimes increase exponentially, respective to increasing input values N. The differences are also affected by the hardware of the computer used, the programming language used, the person writing the functions’ code, the sample values used to calculate the running times and by the method used to calculate the running time (how we obtain the millisecond values found in the driver function). Although the actual running times I have obtained in this work are not identical to the expected ones, we see similar patterns. Keeping in mind that these values are affected by many factors, mentioned above, the results obtained in this report are realistic ones.