CENG2002 - Data Structures - Homework #1

Algorithm Analysis

Mehmet Akif Köz*

*Department of Computer Engineering, Konya Food and Agriculture University, Meram/Konya

REPORT INFO

Keywords:
Fibonacci Series
Recursive Method
Iterative Method
Processing Time
Empirical Analysis
Asymptotic Analysis

ABSTRACT

In this experiment recursive and iterative methods on Fibonacci series are compared with algorithm analysis. The time complexity of the methods is determined by asymptotic analysis. After that the programs had run in real time to determine the execution times. The results show that the recursive method takes more time to execute than the iterative method.

Contents

1. Implementations

1.1 Recursive Implementation

```
int Fib(int n)
{
    if (n < 3)
        return 1;
    return Fib(n - 1) + Fib(n - 2);
}
```

1.2 Iterative Implementation

```
int Fib(int n)
{
    int a = 1, b = 1, temp;
    for (int i = 0; i < n - 2; i++)
    {
        temp = a;
        a = b;
        b += temp;
    }
    return b;
}
```

2. Asymptotic Analysis

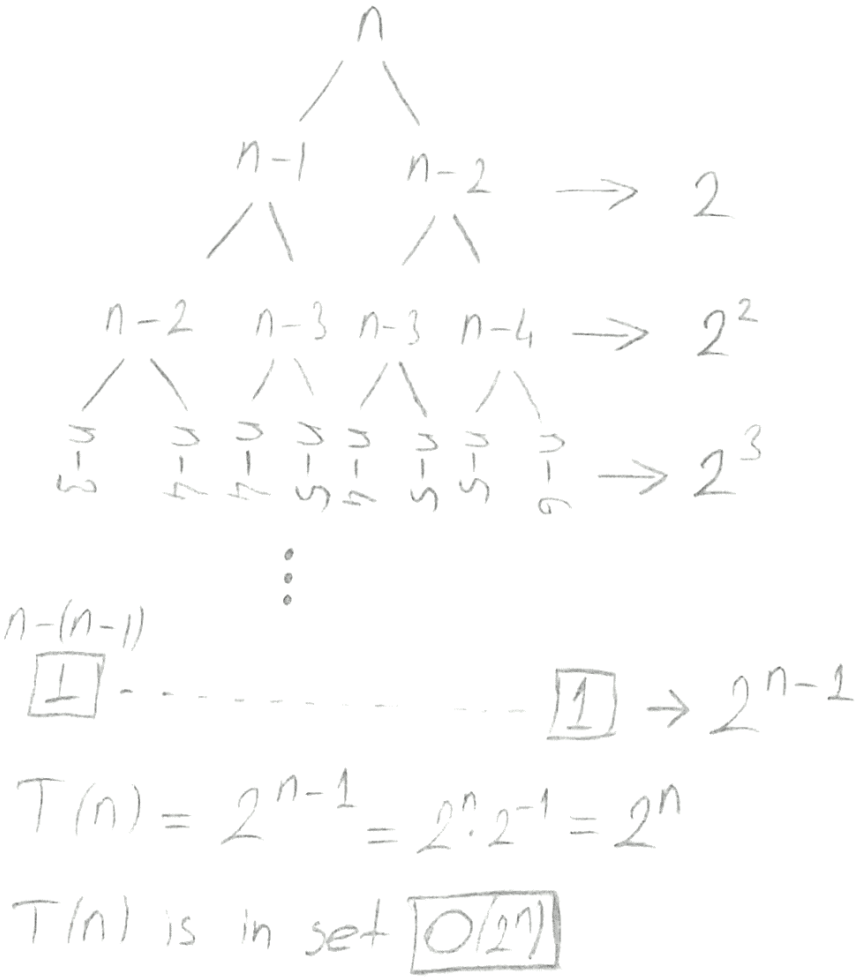
2.1 Recursive

```
int Fib(int n){
    if (n<3)
        return 1;
    return Fib(n-1) + Fib(n-2); }
```

$T(1) = 1$

$T(2) = 1$

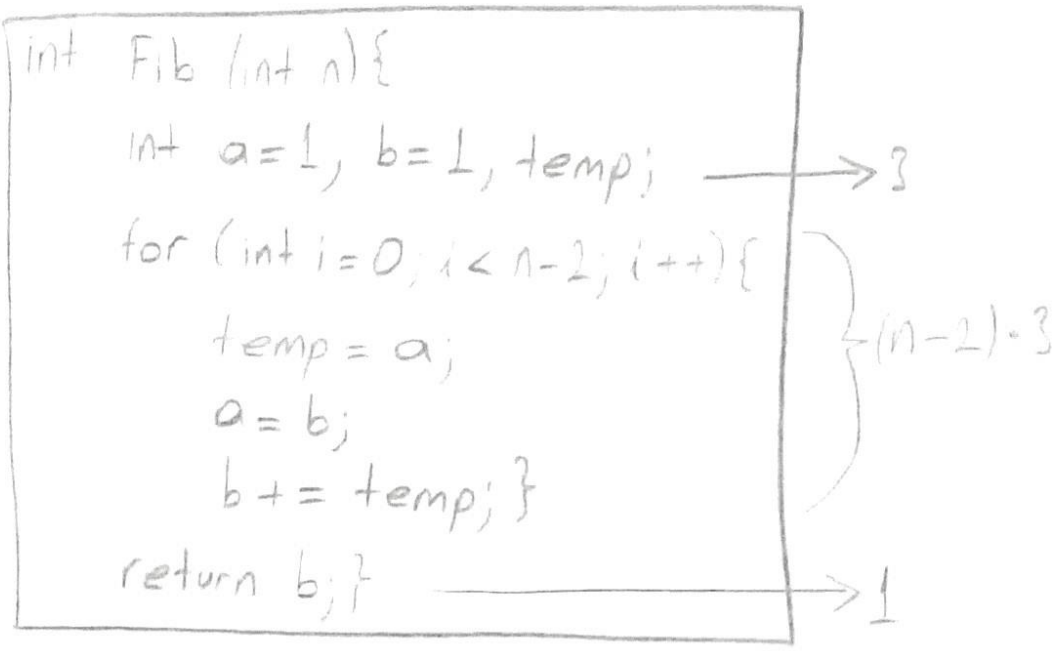
$T(n) = T(n-1) + T(n-2)$



$T(n) = 2^{n-1} = 2^n \cdot 2^{-1} = 2^n$

$T(n)$ is in set $\boxed{O(2^n)}$

2.2 Iterative



$$T(n) = \underbrace{3}_{\#2} + \underbrace{(n-2)\cdot 3}_{\text{RULE \#2}} + \underbrace{1}_{\#2}$$

$$T(n) = \underbrace{1 + (n-2)}_{\text{RULE \#3}} + 1 \qquad \max(n, -2) = n$$

$$T(n) = \underbrace{1 + n}_{\text{RULE \#3}} + 1 \qquad \max(1, n) = n$$

$$T(n) = \underbrace{n + 1}_{\text{RULE \#3}} \qquad \max(n, 1) = n$$

$$T(n) = n$$

$T(n)$ is in set $\boxed{O(n)}$

2.3 Simplifying Rules

1. If $f(n)$ is in $O(g(n))$ and $g(n)$ is in $O(h(n))$, then $f(n)$ is in $O(h(n))$.
2. If $f(n)$ is in $O(kg(n))$ for any constant $k > 0$, then $f(n)$ is in $O(g(n))$.
3. If $f_1(n)$ is in $O(g_1(n))$ and $f_2(n)$ is in $O(g_2(n))$, then $f_1(n) + f_2(n)$ is in $O(\max(g_1(n), g_2(n)))$.
4. If $f_1(n)$ is in $O(g_1(n))$ and $f_2(n)$ is in $O(g_2(n))$, then $f_1(n)f_2(n)$ is in $O(g_1(n)g_2(n))$.

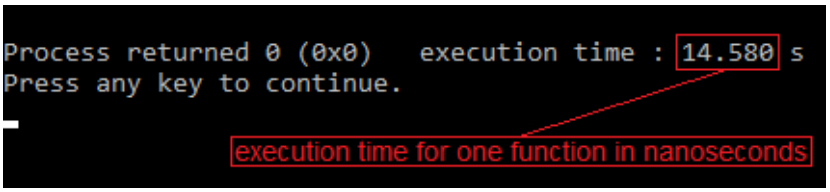
3. Empirical Analysis

A common hardware generally supports around 100 nanoseconds precision. However, to measure the execution time for n = 1, 2, 3, 4, 5, 6, 7, 8, 9 more precision is needed.

Instead of upgrading the hardware, the program could be run many times with a simple for loop.

```
for (int i = 0; i < 1000000000; i++)
    {
        Fib(n);
    }
```

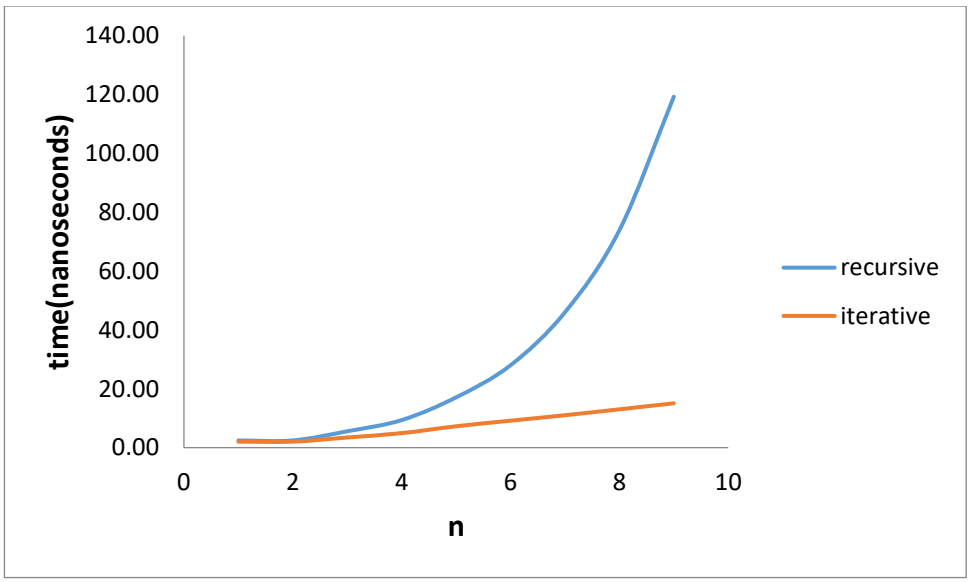
By running the function 10^9 times, nanosecond precision is achieved in a terminal which measures the execution time in seconds.



The execution times are recorded for 20 trials.

3.1 Results

3.1.1 Execution Time Graph



3.1.2 Recursive Results

	n = 1	n = 2	n = 3	n = 4	n = 5	n = 6	n = 7	n = 8	n = 9
T1	2.53	2.44	5.56	9.39	17.05	28.05	46.00	73.85	119.29
T2	2.46	2.46	5.62	9.35	17.04	27.88	46.09	73.84	118.96
T3	2.50	2.45	5.58	9.34	17.16	27.99	46.25	73.65	121.14
T4	2.45	2.46	5.51	9.48	17.18	28.03	45.78	73.49	118.81
T5	2.39	2.41	5.53	9.34	17.12	28.11	45.89	73.92	118.86
T6	2.42	2.49	5.50	9.39	17.16	28.07	45.29	73.87	118.98
T7	2.39	2.45	5.62	9.44	17.06	27.42	45.91	73.68	118.68
T8	2.47	2.45	5.40	9.38	17.05	28.06	45.87	73.69	119.39
T9	2.44	2.39	5.60	9.33	17.04	28.08	46.01	74.12	119.00
T10	2.30	2.43	5.45	9.33	17.08	28.05	45.79	73.75	119.67
T11	2.45	2.47	5.56	9.34	17.13	28.16	47.31	73.67	118.97
T12	2.45	2.46	5.60	9.38	17.01	27.95	46.22	73.87	119.91
T13	2.48	2.53	5.59	9.34	17.02	27.94	45.98	73.94	119.03
T14	2.44	2.48	5.54	9.39	17.12	28.04	45.95	74.19	118.98
T15	2.40	2.47	5.56	9.39	17.01	28.03	45.98	74.07	119.43
T16	2.46	2.44	5.57	9.34	17.08	28.18	45.96	73.79	118.76
T17	2.40	2.38	5.69	9.32	17.06	27.80	45.92	74.04	119.72
T18	2.45	2.42	5.54	9.34	17.13	27.97	46.15	73.60	118.97
T19	2.34	2.50	5.59	9.36	17.10	27.83	45.37	74.20	119.26
T20	2.42	2.40	5.51	9.41	17.09	28.14	46.01	73.79	119.40
avg	2.43	2.45	5.56	9.37	17.08	27.99	45.99	73.85	119.26

3.1.3 Iterative Results

	n = 1	n = 2	n = 3	n = 4	n = 5	n = 6	n = 7	n = 8	n = 9
T1	2.01	2.04	3.47	4.92	7.26	9.20	11.06	13.02	15.07
T2	2.04	2.05	3.43	4.95	7.25	9.16	11.00	13.05	15.13
T3	2.02	2.04	3.46	4.96	7.28	9.24	11.03	13.02	15.10
T4	2.03	2.04	3.43	4.93	7.29	9.15	11.00	13.12	15.10
T5	2.03	2.05	3.45	4.94	7.29	9.17	10.99	13.08	15.09
T6	2.03	2.05	3.43	4.91	7.27	9.16	11.02	13.04	15.11
T7	2.02	2.04	3.45	4.94	7.24	9.20	11.06	13.03	15.09
T8	2.03	2.02	3.45	4.92	7.25	9.19	11.02	13.04	15.11
T9	2.02	2.04	3.42	4.95	7.25	9.17	11.04	12.99	15.04
T10	2.03	2.05	3.45	4.95	7.26	9.15	11.06	13.04	15.07
T11	2.03	2.05	3.45	4.93	7.27	9.15	11.04	12.99	15.08
T12	2.03	2.04	3.45	4.94	7.29	9.12	11.00	12.99	15.04
T13	2.03	2.05	3.44	4.92	7.26	9.15	11.03	13.01	15.04
T14	2.01	2.04	3.44	4.95	7.26	9.19	10.99	13.02	15.02
T15	2.03	2.04	3.46	4.94	7.25	9.15	11.06	13.00	15.09
T16	2.02	2.02	3.43	4.94	7.30	9.14	11.06	13.02	15.02
T17	2.03	2.03	3.45	4.95	7.25	9.15	11.00	13.01	15.08
T18	2.03	2.02	3.44	4.95	7.26	9.13	10.97	13.01	15.00
T19	2.01	2.03	3.45	4.92	7.27	9.13	10.95	13.05	15.05
T20	2.03	2.04	3.44	4.93	7.28	9.16	11.03	12.97	15.05
avg	2.03	2.04	3.44	4.94	7.26	9.16	11.02	13.02	15.07

4. Conclusion

Asymptotic analysis showed that the recursive method has the complexity of O(2^n) and iterative method has O(n). This means on an empirical analysis the recursive method would have exponential growth and iterative have linear growth. Also iterative is faster than recursive. Results of the empirical analysis support these arguments.

To summarize, algorithm analysis is critical to determine the efficiency of an algorithm. Asymptotic analysis is practical to perform but the empirical analysis is not.

5. References

https://stackoverflow.com/questions/15047116/an-iterative-algorithm-for-fibonacci-numbers

https://stackoverflow.com/questions/2607263/how-precise-is-the-internal-clock-of-a-modern-pc

https://youtu.be/AC7KXe4vmgk