A Study on Fibonacci Series Generation Algorithms

Conference Paper · January 2016

DOI: 10.1109/ICACCS.2016.7586379

2 authors, including:

S H Shabbbeer Basha
Indian Institute of Information Technology Sri City
5 PUBLICATIONS **12** CITATIONS

SEE PROFILE

A study on Fibonacci series generation algorithms

S.M Farooq,
MITS Madanapalle,
Chittor, AP, India
shaikfaroq@gmail.com

S.H Shabbeer Basha,
MITS, Madanapalle,
Chittor, AP, India
shabbeermtech@gmail.com

***Abstract*— many algorithms proposed to generate Fibonacci series introduced by a 12th century Italian mathematician Leonardo Bonacci [1]. The study paper gives insight into three different Fibonacci series generation algorithms. This paper compares and contrasts three different algorithms namely LINEAR_FIB, EXPO_FIB and MATRIX_FIB. Time complexities of these algorithms are computed. The results of our study show that the three algorithms gives linear, exponential and logarithmic time complexities. Finally we discussed application areas of Fibonacci numbers which are generated by the algorithms.**

Keywords—Linear; recursive; dynamic programming; logarithmic time complexity.

I. INTRODUCTION

In mathematics, Fibonacci numbers or series are the numbers in the order 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144 (or in modern usage 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89) [2]. By definition, the first two numbers in the fibonacci sequence are either 1 and 1 or 1 and 0. Depending on the chosen starting values the next number is generated by adding the previous two numbers [2]. In mathematical form,

$$F_n = F_{n-1} + F_{n-2}, \text{ Where } F_1 = 0, F_2 = 1$$

Leonardo Bonacci (1170 – 1250) famous western philosopher who popularized hindu-arabic numerical system to the western world. Fibonacci is an Italian mathematician written a famous book titled "Liber Abaci (1202 AD)" (Book of Calculation) on computation. In Liber Abaci, he solved a problem of the growth of rabbits based on some assumptions. The resultant was a sequence of numbers later known as Fibonacci numbers [1].

The study paper presents three different Fibonacci series generation algorithms and computes their time complexities. The three algorithms are LINEAR_FIB, EXPO_FIB and MATRIX_FIB. The respective mathematical derivations are shown for comparison at the last. Finally results are generated to prove that MATRIX_FIB algorithm generates Fibonacci series numbers in logarithmic time when comparing to LINEAR_FIB and EXPO_FIB which generates the numbers in linear and exponential time. Finally the papers presents the applications of Fibonacci numbers. The applications includes barcodes coloring, to analyze Fibonacci heap data structure, these numbers are important in computational analysis of Euclid's algorithm, and also used to analyze Fibonacci heap data structure .

[978-1-4673-9206-8/16/$31.00 ©2016 IEEE]

II. LINEAR_FIB ALGORITHM

Dynamic Programming is an algorithmic paradigm that solves a given complex problem by breaking it into sub problems and stores the results of sub problems to avoid computing the same results again. Unlike Divide and Conquer rule, Dynamic Programming combines solutions to sub-problems. Dynamic Programming is mainly used when solutions of same sub problems are needed again and again [2]. In dynamic programming, computed solutions to sub problems are preserved so that they don't have to re-compute. So Dynamic Programming is not useful when there are no common (overlapping) sub problems because there is no point storing the solutions if they are not needed again. The Fibonacci series can be generated by storing previous results and adding the next value to it. The First algorithm of Fibonacci series generation is LINEAR_FIB. Input to the algorithm is n, a positive integer. Let F is an array to store the result Fibonacci series numbers. Depending on the input n value either 1 or 0 it returns the respective number as output. If the n value is other than 0 or 1 then the array F is initialized with 0 and 1 values in the starting locations. Then the algorithm repeats (iterative) the loop by generating Fibonacci series numbers in the array F.

```
ALGORITHM LINEAR_FIB
LINEAR_FIB (n)
{
    if n = 0 or n = 1
        return 1
    else
        F[1] = 0, F[2] = 1
        for i = 3 to n do
                F[i] = F[i − 1] + F[i − 2]
        return F
}
```

The algorithm LINEAR_FIB, takes approximately n computations (addition operations) for the input value n. so the time complexity can be defined as the number of core computations in the algorithm [2]. The algorithm worst case time complexity is O(n) for the input size n.

III. EXPO_FIB ALGORITHM

The next algorithm is EXPO_FIB. The input to the algorithm is n , a positive integer. Instead of iteratively adding the generated result with the previous value in the series, the EXPO_FIB

algorithm recursively calls to the same function by passing n-1 and n-2 values to generate the Fibonacci series.[3]

ALGORITHM EXPO_FIB

$EXPO\_FIB(n)$

{

if $n = 0$ or $n = 1$

return n

else

return $EXPO\_FIB(n-1) + EXPO\_FIB(n-2)$.

}

Lower bound (Best case time complexity)

To compute time complexity of an EXPO_FIB algorithm, the time taken to compute Fibonacci value n is $T(n)$ which equals to time taken for EXPO_FIB (n-1) and EXPO_FIB (n-2) which are $T(n-1)$ and $T(n-2)$ respectively. The time taken for addition, condition checking is assumed as constant time. Then the recursive relation is,

$$T(n) = \begin{cases} 1, & n < 2 \\ T(n-1) + T(n-2), & n \geq 2 \end{cases}$$

By considering lower bound (best case time complexity) for the function $T(n)$,

$$T(n) \geq 2^* T(n-2)$$
$$\geq 2^* \ 2^* T(n-4)$$

[By substitution method]

$$\geq 4^* T(n-4)$$
$$\geq 4^* 2^* T(n-6)$$
$$\geq 8^* T(n-6)$$

The computation goes on to get generalized term by finding Kth term,

$$\geq 2^k * T(n-2*k)$$

[Kth term]

Now finding k value by equating $n - 2 * k$ to 0

$$n - 2 * k = 0, k = \frac{n}{2}$$

$$\geq 2^{\frac{n}{2}} * T(0)$$

$$\geq 2^{\frac{n}{2}} \ [T(0) = 1]$$

The lower bound for the function $T(n) \geq 2^{\frac{n}{2}}$

Upper bound (Worst case time complexity)

By taking the upper bound (worst case time complexity) for the function $T(n)$.

$$T(n) \leq 2 * T(n-1)$$
$$\leq 2 * 2 * T(n-2)$$

[By substitution method]

$$\leq 4 * T(n-2)$$
$$\leq 4 * 2 * T(n-3)$$
$$\leq 8 * T(n-3)$$
$$\leq 2^k * T(n-k)$$ [Kth term]
$$n - k = 0, k = n$$
$$\leq 2^n * T(0)$$
$$\leq 2^n \ [T(0) = 1]$$

The upper bound for the function $T(n) \leq 2^n$

IV. MATRIX_FIB ALGORITHM

Another method to generate Fibonacci number is matrix exponentiation. The method uses a simple trick matrix multiplication [5] by using the following trick.

Let A be the matrix of the form $A[2][2] = \{(1,1), (1,0)\}$

$$Let\ F_0 = 0, F_1 = 1$$

$$The\ Matrix\ A = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^n = \begin{bmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{bmatrix}$$

$$for\ n = 2, A^2 = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}, here\ F2 = 1$$

$$for\ n = 3, A^3 = A^2 * A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} * \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 3 & 2 \\ 2 & 1 \end{bmatrix}, here\ F3 = 3$$

ALGORITHM MATRIX_FIB (n) //driving function

{

A :={ (1,1),(1,0) }.

if n is 0 or 1 return n.

CALL EXPONENT_MATRIX (A,n-1).

return A[1][1].

}

EXPONENT_MATRIX (A,n)

{

if n >1 then .

CALL EXPONENT_MATRIX (A,n/2).

A:= A*A. // Multiplication of 2X2 Matrix.

if n is odd then

A: =A*{(1, 1), (1, 0)}

}

$$F_n = A^{n-1} * (1, 1)$$

To calculate F_n, we should calculate A^{n-1} matrix and take the first element of the first line. The MATRIX_FIB algorithm uses 2X2 identity matrices A of the form shown. The MATRIX_FIB Algorithm generates Fibonacci numbers in the matrix first row and first column of the matrix A. The algorithm considers both cases of being n value as even and odd numbers and generates Fibonacci numbers by calling EXPONET_MATRIX function. The algorithm still reduces

the time complexity. The logic is to use a tree form to compute matrix multiplication.

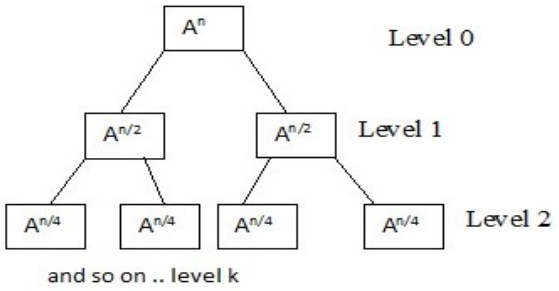

Fig 1. Tree Structure Computation

$$A^n = A^{n/2} * A^{n/2}$$
$$A^{n/2} = A^{n/4} * A^{n/4} \text{ And so on.}$$
$$for\ n = 4, A^4 = A^2 * A^2, A^2 = A * A.\ (n\ is\ even)$$

The recurrence relation for the MATRIX_FIB algorithm is,

$$T(n) = \begin{cases} T\left(\frac{n}{2}\right) + C, & n > 2 \\ 1, & n = 1\ or\ n = 2 \end{cases}$$

The time taken to compute T(n) is equals to the time to evaluate T(n/2) plus constant time (C) if the n value is greater than 2 other wise 1 some constant time to evaluate matrix multiplication of 2X2 matrix.

By using substitution method,

$$T(n) = T\left(\frac{n}{2}\right) + C$$
$$= T\left(\frac{n}{4}\right) + 2 * C$$
$$= T\left(\frac{n}{8}\right) + 3 * C$$
$$= T\left(\frac{n}{16}\right) + 4 * C$$
$$= T\left(\frac{n}{2^k}\right) + k * C \quad [\text{ Kth Term }]$$

Equating $\frac{n}{2^k} = 1, then\ n = 2^k, k = log_2(n)$

$$= T(1) + log_2(n) * C$$
$$= 1 + log_2(n)$$

Here the multiplication of 2X2 Matrix takes constant amount of time order of 1. Finally we get O($log_2\ n$) as time complexity.

V. COMPARISION

TABLEI. COMPARISON OF ALGORITHMS

S.No.	Algorithm	Time complexity	Behavior
1.	LINEAR_FIB	O(n)	Linear
2.	EXPO_FIB	O(2^n)	Exponential
3.	MATRIX_FIB	O($log_2\ n$)	Logarithmic

VI. RESULTS

The above linear, exponential and logarithmic time Fibonacci number generation algorithms are implemented in c programming language. The execution times of respective programs are captured using time command in UNIX environment. The time command in UNIX operating system gives statistics of execution time of the program run. These statistics consists of elapsed real time between invocation and termination of a program. The real time value is like turnaround time which includes waiting time plus burst time. The user time gives total amount of CPU Seconds a process spends in user mode. The System time gives the total amount of CPU seconds a process spends in kernel mode.
The execution time values of linear, exponential and logarithmic programs are generated and plotted.

TABLE II.

Sample input (n) and its computational values (T) For LINEAR_FIB algorithm

S.No	Input n	T (ms)
1	1	1
2	300	2
3	500	3
4	600	4
5	700	5
6	800	6
7	850	7
8	900	8
9	950	9
10	1050	10

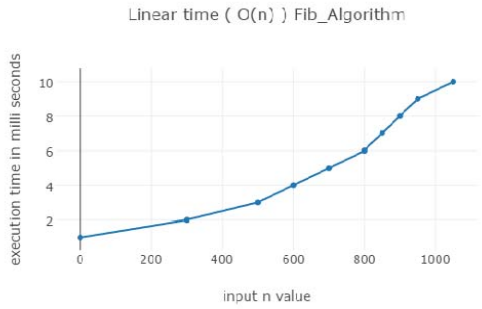

Fig. 2 results and plot of linear algorithm (PLOT I)

The first plot is about LINEAR_FIB program behavior. The input n values in the range 1-1000 and their respective execution time in mille seconds are plotted. The results of UNIX time command shows that for increasing values of n in terms of hundreds the behavior of execution time T of the program is linearly increasing. The second plot is about EXPO_FIB program behavior. The input n values in the range 1-50 and their respective execution time T in milli seconds are plotted.

TABLE III.

Sample input(n) and its computational values(T) for EXPO_FIB algorithm

S.No	Input n	T (ms)
1	1	0.001
2	25	0.003
3	30	0.023
4	35	0.2434
5	40	2.6891
6	45	29.852
7	50	533.66
8	55	--

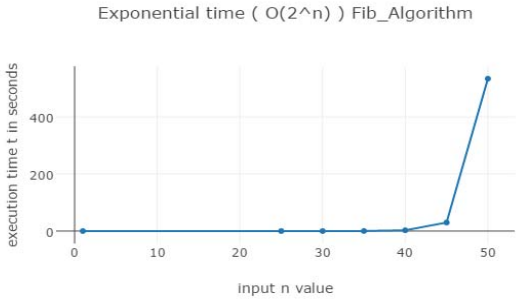

Fig. 3 results and plot of exponential algorithm (PLOT II)

The results of UNIX time command shows that for the values 1-35 the program is executed in constant time. for n value greater than 35 it running time shows exponential behavior with respect to n. for n=45, the program is executed in half minutes (29.852 seconds). For n=50 the program executed in 8.89 minutes which shows exponential behavior.

TABLE IV.

Sample input(n) and its computational values(T) for MATRIX_FIB algorithm

S.No	Input n	T (ms)
1	1	0.001
2	11500	0.002
3	22500	0.003
4	23601	0.003
5	23602	Inf

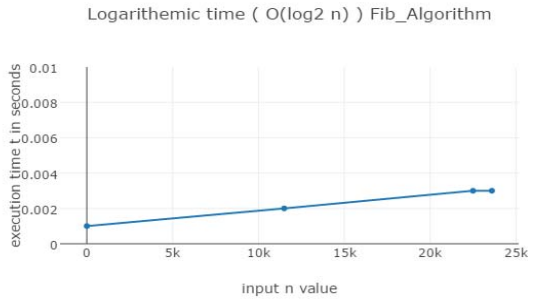

Fig. 4 results and plot of logarithmic algorithm (PLOT III)

The third plot is about MATRIX_FIB program behavior. The input n values in the range 1-25000 and their respective execution time in seconds are plotted. The results of UNIX time command shows that for the values 1- -23000 the running

time of a program logarithmically increasing which is almost flat curve in the plot.

VII. CONCLUSION

The study paper compares time complexities three different algorithms that generate age old Fibonacci numbers. The practical results that are generated using UNIX time command and plotted using online web portal plot.ly. The results prove the linear, exponential and logarithmic behavior of algorithms. Further Fibonacci numbers used in color barcodes [4], to analyze Fibonacci heap data structure, these numbers are important in computational analysis of Euclid's algorithm [5], and also used to analyze Fibonacci heap data structure [6].

VIII. REFERENCES:

[1]. The Fabulous Fibonacci Numbers - Posamentier & Lehmann, 2007, pp.26-27.

[2]. Fundamentals of Computer Algorithms, Ellis Horowitz, Sartaj Sahni, S Rajasekaran, University Press, Second Edition (2008).

[3]. stojmenovic, "Recursive algorithms in computer science courses: Fibonacci numbers and binomial coefficients ", IEEE transactions on education, Volume 43, Issue 3, 2000, pp.273-276.

[4]. Agaian, "Generalized Fibonacci numbers and applications" in proceedings of Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference, 3484-3488.

[5]. Knuth, Donald E (1997), The Art of Computer Programming, 1: Fundamental Algorithms (3rd ed.), Addison–Wesley, p. 343, ISBN 0-201-89683-4 [6] "Coverage problems in Wireless Ad-hoc Sensor Networks" Seapahn Meguerdichian, Farinaz Koushanfar, Miodrag Potkonjak, Mani B. Srivastava IEEE INFOCOM 2001.

[6]. Pettie, S. ; Max Planck Inst. fur Informatik, Germany Towards a final analysis of pairing heaps ISBN: 0-7695-2468-0.