Journal of Information Technology and Computer Science
Volume 1, Number 1, 2016, pp. 14 – 27
Journal Homepage: www.jitecs.ub.ac.id

Review: A State-of-the-Art of Time Complexity (Non-Recursive and Recursive Fibonacci Algorithm)

Imam Cholissodin[1], Efi Riyandani[2]
[1,2]Faculty of Computer Science, University of Brawijaya, Indonesia
[1]imamcs@ub.ac.id, [2]efiriyandani@gmail.com

Received 21 February 2016; received in revised form 18 March 2016; accepted 25 March 2016

Abstract. Solving strategies in the computation the time complexity of an algorithm is very essentials. Some existing methods have inoptimal in the explanations of solutions, because it takes a long step and for the final result is not exact, or only limited utilize in solving by the approach. Actually there have been several studies that develop the final model equation Fibonacci time complexity of recursive algorithms, but the steps are still needed a complex operation. In this research has been done several major studies related to recursive algorithms Fibonacci analysis, which involves the general formula series, begin with determining the next term directly with the equation and find the sum of series also with an equation too. The method used in this study utilizing decomposition technique with backward substitution based on a single side outlining. The final results show of the single side outlining was found that this technique is able to produce exact solutions, efficient, easy to operate and more understand steps.
Keywords: *Time Complexity, Non-Recursive, Recursive, Fibonacci Algorithm*

1 Introduction

The time complexity of an algorithm can be solved with the approach that is usually performed by analysis of the syntax of an algorithm that begins with created a initial function use the symbol sigma with a specified lower bound and upper bound value (for non-recursive), or does not use the symbol sigma (for recursive). And from the initial functions that contain symbol sigma will be simplified into a function that can be directly calculated result of its complexity without must go through an iterative process or another similar mode of operation like that. The way of that it can be a solution to create formula as a model equation for the time complexity of any algorithm.

Some techniques that are typically used to solve the initial function is a Backward Substitution, Master Theorem, and other. Backward Substitution is outlining iteratively until get and can identifying the patterns of result including into the specific series. Backward Substitution is more priority to outlining any side that exist from initial equation or initial functions (as in outlining equation (20) with both side). Each side is outlining or elaboration iteratively again until meet the best case conditions, that it involve needs a long time and difficult to solve too. In fact, for each side of elaboration operations, these function appeared mostly side by side identical form of decomposition

operation on one or several levels from previous steps or the next steps analysis, so it's actually can be simplified.

It can be seen that the decomposition of the initial function at the solving measured of time complexity to the algorithm Fibonacci is entered in the form of recursive analysis. In a previous study, Binet is a scientist math has successfully proved one of number theory very unique (number theory), the theory for solving the order of the Fibonacci sequence (Fibonacci series) directly with an equation non-recursive, and also Lucas (1870-1871) who proposed the theory of numbers associated with the series called Lucas sequence [10][11][8] such as those mentioned in the equation (1). As a consequence, the series is easy to calculate and understood by other researchers widely. But for measured time complexity of Fibonacci numbers are still rare that connects the pseudo code that is used from the many call recursive functions and input size of it's after elaborated or described by the theory of Binet, and most previous studies as in [9] is not mentioned and explained clearly and detail.

Odendahl (2016) has made measured time complexity of Fibonacci algorithms, but using the approach, namely by equating or change the value of side of the fewer steps (input size is smaller) with a side of the steps more or deeper (input size is larger) [9]. As a consequence the value of the final result of time complexity are inexact, or merely restrict the actual solution and can be ensured not optimal, that is equal to $O(2^n)$. The first problem is how to resolve the Fibonacci time complexity of the algorithm using the proposed method to produce the exact solution. But, until now the existing methods are still difficult to understand how it can be solve because the explanations are not simple. The second problem that arises is when using decompose using more than one side, two side or all sides, it will result in inefficient in terms of speed to get the value of the same solution. Therefore, in this research, the proposed technique, parsing one side and the other side with identical shape, will be equated and need not be described more depth. In this case, this step is a special treatment that really should be done to create a strategy to reach a solution through an approach of the analysis the pattern of results itself. Thus, the end result of completion the time complexity can be faster and also proved to be more simple in the steps of the solution.

2 Fibonacci Number

Fibonacci numbers are a sequence which has a unique form that was discovered by Leonardo of Pisa in Fig. 1, a mathematician from Italia. One of the things that underlie him, it was when he began to be interested in learning the concept of the number of Arabs who are easier to use for any calculations and more practical, than the Roman numerals from their own nation. He has contributed and been instrumental in introducing the Arabic numerals to the Westerner [1]. And this can be imagined how difficult the west in doing any calculations, previously while being ignorant of Arabic numerals. In addition, the Fibonacci number patterns are also known in some form that exists in nature, and when identified, it has been known, here there are, the value of the unique scale called the golden ratio ($\phi = 1 + \phi^{-1} = (1 + \sqrt{5})/2 \approx 1.618$), it is a key digits, which is able to generate Fibonacci sequence that is very interesting to learn more deeply [2][3].

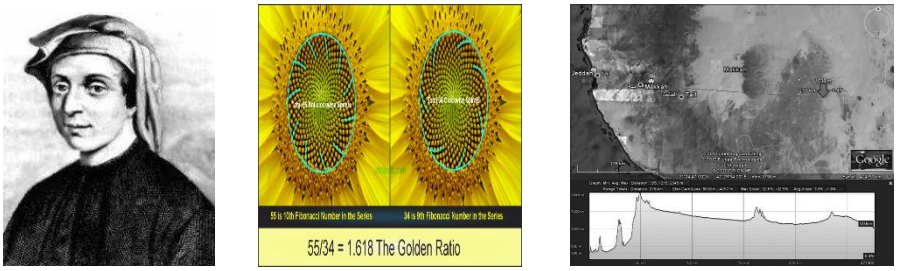


Fig. 1. Leonardo of Pisa, Fibonacci series on living beings and the position on the earth.

A non-recursive and recursive method in java programming that can be used to compute the *n*-th term of Fibonacci sequence can be seen in Table 1 [7], and also displayed in the source code of the form of equation Binet's closed-form formula [5]. From the results, it is displaying the Fibonacci sequence that has a little different from its initial value, ie there that starts at 0 and there are starts at 1.

Table 1. Source Code of Fibonacci Non-Recursive and Recursive

Non-Recursive		Recursive
Iterative	Non-Iterative (Binet's closed-form)	

int fibo(int n) {
if (n <= 1) {
return 1;
} else {
int oldfib = 1, fib = 1;
for (int i = 2; i <= n; i++) {
int newfib = oldfib + fib;
oldfib = fib;
fib = newfib;
}
return fib;
}
}
 	
Output: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ..
	static int fibo(int n){
	int a,b,c;
	a=(int)
	Math.pow((1+
	Math.sqrt(5)),n);
	b=(int)
	Math.pow((1-
	Math.sqrt(5)),n);
	c=(int)
	(Math.pow(2, n)*
	Math.sqrt(5));
	return ((a-b)/c);
	}
	
	Output: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ..
		int fibo(int n) {
		if (n<=1)
		return 1;
		else
		return fibo(n-1) + fibo(n-2);
		}
		
		Output: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ..

The more interesting is finding Fibonacci numbers in the shape of the earth, such as the research that was conducted by Akhtaruzzaman and Amir (2011) which states that part of the position of the Earth, that has a value of the golden ratio, which exists in Mecca [4]. The shape of the Fibonacci sequence could be constructed clearly by initializing the initial two values, 0 and 1 as the first and second term.

Each the *n*-th term have the basic rules, namely $U_n = U_{n-1} + U_{n-2}$. However, in many previous types of research, U_n have been using an equation directly without any conditions recursively in equation [5].

Series : 0,1,1,2,3,5,8,13,21,34,55,….

$$U_n = \frac{(1+\sqrt{5})^n - (1-\sqrt{5})^n}{2^n\sqrt{5}} = \frac{\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n}{\left(\frac{1+\sqrt{5}}{2} - \frac{1-\sqrt{5}}{2}\right)}$$
$$= \frac{(\phi)^n - (\psi)^n}{\phi - \psi} \quad (1)$$

Where, $\alpha^2 - \alpha - 1 = 0$,

So that $\alpha_{1,2} = \frac{-b \pm \sqrt{b^2-4ac}}{2a} = \frac{1 \pm \sqrt{(-1)^2-4.1.(-1)}}{2.1} = \frac{1 \pm \sqrt{5}}{2}$

$\phi = \alpha_1 = \frac{1+\sqrt{5}}{2}$ and $\psi = \alpha_2 = \frac{1-\sqrt{5}}{2}$

As well as applicable $\phi = 1 - \psi$

Where, $\phi^2 - \phi - 1 = 0, \phi^2 = \phi + 1, \phi = \frac{\phi+1}{\phi}$

Series ϕ^i is $\phi^0, \phi^1, \phi^2, \dots, \phi^n = 1, \phi, \left(\frac{\phi+1}{\phi}\right)\phi, .., \quad i = 0, 1, \dots, n$

$= 1, \phi, (\phi + 1), (\phi + 1)\phi, \dots = 1, \phi, (\phi + 1), (\phi^2 + \phi), ..$
$= 1, \phi, (\phi + 1), \big((\phi + 1) + \phi\big), \dots = 1, \phi, (\phi + 1), (2\phi + 1), (2\phi + 1)\phi, ..$
$= 1, \phi, (\phi + 1), (2\phi + 1), (2\phi^2 + \phi) \dots = 1, \phi, (\phi + 1), (2\phi + 1), (2(\phi + 1) + \phi), ..$

$= 1, \phi, (\phi + 1), (2\phi + 1), (3\phi + 2), (3\phi + 1)\phi, \dots = 1, \phi, (\phi + 1), (2\phi + 1), (3\phi + 2), (3\phi^2 + 2\phi), ..$
$= 1, \phi, (\phi + 1), (2\phi + 1), (3\phi + 2), (3(\phi + 1) + 2\phi), \dots = 1, \phi, (\phi + 1), (2\phi + 1), (3\phi + 2), (5\phi + 3), ..$
$= 1, \phi, (\phi + 1), (2\phi + 1), (3\phi + 2), (5\phi + 3), (8\phi + 5), (13\phi + 8), (21\phi + 13), (34\phi + 21), \dots$

Series ψ^i is $\psi^0, \psi^1, \psi, \dots, \psi^n = 1, \psi, \left(\frac{\psi+1}{\psi}\right)\psi, .., \quad i = 0, 1, \dots, n$
$= 1, \psi, (\psi + 1), (2\psi + 1), (3\psi + 2), (5\psi + 3), (8\psi + 5), (13\psi + 8), (21\psi + 13), (34\psi + 21), \dots$

Series $\phi^i - \psi^i$ where $i = 0,1, \dots, n$
$= 1 - 1, \phi - \psi, (\phi + 1) - (\psi + 1), (2\phi + 1) - (2\psi + 1), (3\phi + 2) - (3\psi + 2), \dots$
$= 0, \phi - \psi, \phi - \psi, 2(\phi - \psi), 3(\phi - \psi), 5(\phi - \psi), 8(\phi - \psi), 13(\phi - \psi), 21(\phi - \psi) \dots$

If series $\phi^i - \psi^i$ divided by $(\phi - \psi)$, it will be the Fibonacci sequence.

$$\frac{\phi^i-\psi^i}{\phi-\psi} = \frac{0}{\phi-\psi}, \frac{\phi-\psi}{\phi-\psi}, \frac{\phi-\psi}{\phi-\psi}, \frac{2(\phi-\psi)}{\phi-\psi}, \frac{3(\phi-\psi)}{\phi-\psi}$$

$$\frac{\phi^i-\psi^i}{\phi-\psi} = 0,1,1,2,3,5,8,13, \ldots.$$

$$U_n = \frac{\left(1+\sqrt{5}\right)^n - \left(1-\sqrt{5}\right)^n}{2^n\sqrt{5}} = \frac{\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n}{\left(\frac{1+\sqrt{5}}{2} - \frac{1-\sqrt{5}}{2}\right)} = \frac{(\phi)^n - (\psi)^n}{\phi-\psi}$$

$$S_n = \sum_{i=0}^{n} \frac{\phi^i-\psi^i}{\phi-\psi} = \frac{\phi^0-\psi^0}{\phi-\psi} + \sum_{i=1}^{n} \frac{\phi^i-\psi^i}{\phi-\psi} = 0 + \sum_{i=1}^{n} \frac{\phi^i-\psi^i}{\phi-\psi} = \sum_{i=1}^{n} \frac{\phi^i-\psi^i}{\phi-\psi}$$

By using the concept of geometric series, then obtained the summation sequence of Fibonacci numbers to the *n*-th (S_n) with completion as below:

$$S_n = \frac{1}{\phi-\psi}\left(\sum_{i=1}^{n} \phi^i - \sum_{i=1}^{n} \psi^i\right)$$

$$S_n = \frac{1}{\phi-\psi}\left(\frac{\phi(\phi^n-1)}{\phi-1} - \frac{\psi(\psi^n-1)}{\psi-1}\right) = \frac{1}{\phi-\psi}\left(\phi^2(\phi^n-1) + \psi^2(1-\psi^n)\right)$$

$$S_n = \frac{1}{\sqrt{5}}\left(\left(\left(\frac{1+\sqrt{5}}{2}\right)^2\left(\left(\frac{1+\sqrt{5}}{2}\right)^n - 1\right)\right) + \left(\left(\frac{1-\sqrt{5}}{2}\right)^2\left(1 - \left(\frac{1-\sqrt{5}}{2}\right)^n\right)\right)\right) \quad (2)$$

3 Time Complexity Analysis Non-Recursive and Recursive Case

Time Complexity Analysis was used to calculate the time of computing an algorithm based on the number of iteration in the basic operation. So, researcher it can choose the better, which algorithm that should be used among other things based on the value of the time complexity according to the input size of data. Before discussed in more detail, related to the complexity of recursive algorithms, here will be described and presented other examples related to the completion of the algorithm non-recursive very easy to understand together, as a fundamental illustration of how the analysis works, the algorithm is Sequential Search, and Unique Element [6]. There are three kinds of time complexity of the non-recursive algorithm, namely Best Case, Worst Case and Average Case.

Sequential Search:

```
i ← 0
while i < n    and A[i] ≠ K do
        i ← i + 1
if    i < n    return i
else return    -1
```

a. Best case:

$$C(n) = \sum_{i=0}^{n-1} 1 \quad (3)$$

Because the definition of the base case is the value of *K* is found in the first element in the array *A*, then the equation (3) at the upper bound of the index *i* can be equated to the value of lower bound, namely (*n*-1) = 0.

$$C(n) = \sum_{i=0}^{n-1} 1 = \sum_{i=0}^{0} 1 \quad (4)$$

And to make it easier in solving equation (4), because it is seen that in the sigma, there are no indexes that are being run in sigma, so it can be modified lower and upper values, by giving the same treatment, such as lower and upper bound respectively their added by value +1.

$$C(n) = \sum_{i=0}^{n-1} 1 = \sum_{i=0}^{0} 1 = \sum_{i=0+1}^{0+1} 1 = \sum_{i=1}^{1} 1 = 1 \quad (5)$$

Result of Best Case in equation (5) will be used a basis for calculating the Best Case of Unique Element algorithm, which is known the procedure as follows

Unique Element:

```
for i ← 0 to n − 2 do
        for j ← i + 1 to n - 1 do
                If A[i] = A[j] return false
return true
```

$$C(n) = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1 = \sum_{i=0}^{0} \sum_{j=i+1}^{i+1} 1 = \sum_{i=0}^{0} \sum_{j=i+1-i}^{i+1-i} 1 = \sum_{i=0}^{0} \sum_{j=1}^{1} 1$$
$$= \sum_{i=0}^{0} 1 = \sum_{i=0+1}^{0+1} 1 = \sum_{i=1}^{1} 1 = 1 \quad (6)$$

b. Worst Case

Definition of Worst Case on Sequential Search is the value of *K* is found in the last element in the array *A*. In $C(n) = \sum_{i=0}^{n-1} 1$ all iteration should be run from beginning to end, and it can be modified lower bound and upper bound because in the sigma, no index *i* is being run. At the beginning of the index value, if starting from *i* = 0 is considered less prevalent, and should be be modified into *i* = (0 + 1) so that *i* starting from a value of 1, this is used to easier in calculation. If the lower bound is changed, the upper bound should also be given the same treatment to be changed too, from (*n* - 1) to (*n* - 1 + 1) or can be simply written in a simple form become *n*.

$$C(n) = \sum_{i=0}^{n-1} 1 = \sum_{i=0+1}^{n-1+1} 1 = \sum_{i=1}^{n} 1 = n \quad (7)$$

Step by step to solving Worst Case in equation (7) will be used as the basis for calculating the Worst Case of Unique Element algorithm as follows:

$$C(n) = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1 = \sum_{i=0}^{n-2} \sum_{j=i+1-i}^{n-1-i} 1 = \sum_{i=0}^{n-2} \sum_{j=1}^{n-1-i} 1 = \sum_{i=0}^{n-2} (n-1-i)$$
$$= \sum_{i=0}^{n-2} (n-1-i) \quad (8)$$

Because of the equation (8), it is known that in $C(n)$ there are indices i are being run in sigma, then there should be no modification of the lower and upper bounds. The solution is described by the elaboration until at specific depths, and then identified a pattern, including what the series? and then made simplification process. Thus, in equation (8) has been not contain sigma.

$$\begin{aligned} C(n) &= \sum_{i=0}^{n-2}\sum_{j=i+1}^{n-1} 1 = \sum_{i=0}^{n-2}\sum_{j=i+1-i}^{n-1-i} 1 = \sum_{i=0}^{n-2}\sum_{j=1}^{n-1-i} 1 = \sum_{i=0}^{n-2}(n-1-i) \\ &= \sum_{i=0}^{n-2}(n-1-i) = (n-1-0)+(n-1-1)+\cdots+\big(n-1-(n-2)\big) \\ &= (n-1)+(n-2)+(n-3)+(n-4)+(n-5)+\cdots+(n-1-n+2) \\ &= (n-1)+(n-2)+(n-3)+(n-4)+(n-5)+\cdots+1 \end{aligned} \quad (9)$$

In equation (9), if this analyzed more deeply, the description forms the pattern of arithmetic series with a first term is (n-1), the last term is 1, the number of terms is (n-2 + 1) = (n-1), the reason added by value 1 because the lower bound value starting from zero, and the different for each $U_n - U_{n-1}$ is equal same value, namely -1. Thus, simplified form of equation (9) above, became as below [14]:

$$\begin{aligned} C(n) &= \frac{number\ of\ terms}{2}(first\ term + last\ term) \\ &= \frac{(n-1)}{2}\big((n-1)+1\big) \\ &= \frac{(n-1)}{2}(n) \\ &= \frac{n^2-n}{2} \end{aligned} \quad (10)$$

Based on Table 1, steps to resolve the equation (7) can also be used as a guide for a solution for the calculation of Non-Recursive Fibonacci algorithm.

$$C(n) = \sum_{i=2}^{n} 1 = \sum_{i=2-1}^{n-1} 1 = \sum_{i=1}^{n-1} 1 = n-1 \quad (11)$$

$$C(n) = O(n) \quad (12)$$

c. Average Case

The concept of the Average Case is K found in the middle element of the array A, which can be found in the 2nd element or on one element before the last element. Another explanation, if a value K is found in elements exclude from the first element and the last element, it can be said that the condition includes the Average Case. In the perspective of a simple analysis, ideally, these average conditions exist in the position of an element in the midst Array. But on different case or instances, for example the value in its array is different and K that searched that too is different, then this can be ascertained that the value of K is found not always in the mid of element, or more precisely the solution position will always be random, especially when each value of K is changed. Therefore, the calculation of the complexity of the specific time for the average case condition, consider the value of the weight in the form of probability value, that every found of the value of K at a specific position, for example at the position iteration i-th and i will be multiplied by the value of the probability of a K value when found at the time, namely with number of occurrences is only once, so the

number of events is 1, and the value of this (1) then be divided by the number of all the iteration are possible from the search process, for this matter the value can be taken from the value of Worst case in equation (7) of the Sequential Search algorithm ($C(n) = n$). The first step that should do is to ascertain whether the value of a lower bound for each index has been started from the value of 1. This is because the index value as the value multiplier that have meaning as many operation of iteration that has been (or already) run, so if starting from 0, then it is like no operations of iteration, although it will be multiplied with any big value of probability then the result must equal with zero and not contributes anything towards the calculation of average case, even though the initial value 0, It states the index for the first iteration. Forms formula of time complexity $C(n) = \sum_{i=0}^{n-1} 1$ based on equation (7) will be changed to $C(n) = \sum_{i=0+1}^{n-1+1} 1 = \sum_{i=1}^{n} 1$ as a guide for resolving the time complexity for average case. Here are detailed descriptions of the calculation Average Case for Sequential Search algorithm.

$$C(n) = \sum_{i=1}^{n} \left(counter \frac{1}{C_{worst}(n)} \right)$$
$$= \sum_{i=1}^{n} \left(i \frac{1}{C_{worst}(n)} \right) = \sum_{i=1}^{n} \left(i \frac{1}{n} \right) = \frac{1}{n} \sum_{i=1}^{n} i = \frac{1}{n} \left(\frac{n}{2} (1 + n) \right) = \frac{(1+n)}{2} \quad (13)$$

In equation (13), why *counter* = *i*, because for identifying the number of iteration, so it can be continue to increase value in sequentially up to the value of the magnitude time complexity of the worst case, that if it displayed would generate output like 1, 2, 3, 4, 5, 6, ..., $C_{worst}(n)$.

Step by step to solving Average Case in equation (13) will be used as a guide for calculating the Average Case of Unique Element algorithms, with a viewpoint, whether the same element may be discovered at the position, exclude from the Best Case and Worst Case, and for details of the solution can be viewed as below:

1. Change the lower bound if it not starting from 1, index value should be changed starting from 1, for example, the time complexity
   $C(n) = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1$ in equation (8) can changed into,

   $$C(n) = \sum_{i=0+1}^{n-2+1} \sum_{j=i+1}^{n-1+1} 1 = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} 1 \quad (14)$$

   Because in the equation (14), lower bound at the first sigma notation, value i = 0, changed into i = (0 + 1), so upper bound at the first sigma notation must be changed from (n-2) become ((n-2)+1). And automatically, it will have an effect also in the second sigma, it mean if the value of *i* in the first sigma added by value 1, and because lower bound in the second sigma, valued *j* = (*i* + 1), which states that the value of the index of *j* depends on the value of *i*, so the upper bound in the second sigma should also be added value 1, in order not to change the number of operation of in the iteration, from an early form of the both sigma, before be done the change.
2. In the same way from the equation (14), it will be attempted to be used in the average case solution from Unique Element algorithm.

$$C(n) = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\left(counter\frac{1}{C_{worst}(n)}\right) = \sum_{i=1}^{n-1}\sum_{j=i+1-i}^{n-i}\left(counter\frac{1}{C_{worst}(n)}\right)$$
$$= \sum_{i=1}^{n-1}\sum_{j=1}^{n-i}\left(counter\frac{1}{C_{worst}(n)}\right)$$
$$= \sum_{i=1}^{n-1}\sum_{j=1}^{n-i}\left(\left(j+\left(\frac{(i-1)}{2}\left((n-1)+(n-i)+1\right)\right)\right)\frac{1}{C_{worst}(n)}\right) \quad (15)$$

In equation (15), why variable of *counter* = $\left(j+\left(\frac{(i-1)}{2}\left((n-1)+(n-i)+1\right)\right)\right)$, because for identifying the number of iteration, so it can be continue to increase value in sequentially up to the value of the magnitude of the worst case of time complexity, and if it displayed would generate output 1, 2, 3, 4, 5, 6, ..., $C_{worst}(n)$. Because in the formula of *counter* containing index *j*, then it should in the second sigma unmodified in the lower and upper bound of it, so it must be returned to its initial form, and *j* still must, conditioned and created equal with starting from a value of 1, so when *j* = (*i*+1), then *j* in the second sigma must be minus by *i*, and equation (15) became like below

$$C(n) = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\left(\left((j-i)+\left(\frac{(i-1)}{2}\left((n-1)+(n-i)+1\right)\right)\right)\frac{1}{C_{worst}(n)}\right)$$
$$= \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\left(\left((j-i)+\left(\frac{(i-1)}{2}(2n-i)\right)\right)\frac{1}{\frac{n(n-1)}{2}}\right)$$
$$= \frac{2}{n(n-1)}\sum_{i=1}^{n-1}\left(\sum_{j=i+1}^{n}j+\left(-i+\frac{(i-1)}{2}(2n-i)\right)\sum_{j=i+1}^{n}1\right)$$
$$= \frac{2}{n(n-1)}\sum_{i=1}^{n-1}\left(\frac{(n-i)}{2}(i+1+n)+\left(-i+\frac{(i-1)}{2}(2n-i)\right)\sum_{j=1}^{n-i}1\right)$$
$$= \frac{2}{n(n-1)}\sum_{i=1}^{n-1}(n-i)\left(\frac{(i+1+n)}{2}+\left(-i+\frac{(i-1)}{2}(2n-i)\right)\right)$$
$$= \frac{2}{n(n-1)}\sum_{i=1}^{n-1}\frac{(n-i)}{2}\left((i+1+n)+\left(-2i+\left(i(2n-i)-1(2n-i)\right)\right)\right)$$
$$= \frac{1}{n(n-1)}\sum_{i=1}^{n-1}(n-i)\left(i+1+n-2i+2ni-i^2-2n+i\right)$$
$$= \frac{1}{n(n-1)}\sum_{i=1}^{n-1}(n-i)\left(2ni+(1-n)-i^2\right)$$
$$= \frac{1}{n(n-1)}\sum_{i=1}^{n-1}\left(n\left(2ni+(1-n)-i^2\right)-i\left(2ni+(1-n)-i^2\right)\right)$$
$$= \frac{1}{n(n-1)}\sum_{i=1}^{n-1}\left(2n^2i+n-n^2-ni^2-2ni^2-(1-n)i+i^3\right)$$
$$= \frac{1}{n(n-1)}\sum_{i=1}^{n-1}\left(\left(2n^2-(1-n)\right)i+\left(n-n^2\right)-3ni^2+i^3\right)$$
$$= \frac{1}{n(n-1)}\left(\begin{matrix}\left(2n^2-1+n\right)\sum_{i=1}^{n-1}i+\left(n-n^2\right)\sum_{i=1}^{n-1}1-3n\sum_{i=1}^{n-1}i^2\\ +\sum_{i=1}^{n-1}i^3\end{matrix}\right)$$

$$= \frac{1}{n(n-1)} \begin{pmatrix} \left(2n^2 - 1 + n\right)\frac{(n-1)}{2}\left(1 + (n-1)\right) + \left(n - n^2\right)(n-1) \\ -3n\frac{(n-1)((n-1)+1)(2(n-1)+1)}{6} \\ +\left(\frac{(n-1)}{2}\left(1 + (n-1)\right)\right)^2 \end{pmatrix}$$

$$= \frac{(2n^2-1+n)}{2} + (1-n) - \frac{3n(2n-1)}{6} + \frac{(n-1)n}{4}$$

$$= \frac{(2n^2-1+n)}{2} + (1-n) - \frac{n(2n-1)}{2} + \frac{(n-1)n}{4}$$

$$= \frac{2(2n^2-1+n)}{4} + \frac{4(1-n)}{4} - \frac{2n(2n-1)}{4} + \frac{(n-1)n}{4}$$

$$= \frac{1}{4}\left(2\left(2n^2 - 1 + n\right) + 4(1-n) - 2n(2n-1) + (n-1)n\right)$$

$$= \frac{1}{4}\left(4n^2 - 2 + 2n + 4 - 4n - 4n^2 + 2n + n^2 - n\right)$$

$$= \frac{1}{4}\left(n^2 - n + 2\right) \quad (16)$$

For the fast solution, namely by do the conversion the both sigma notation in equation (15) into a single sigma. It is used to simplify the solution of the time complexity Average Case of Unique Element algorithm based on the viewpoint that discovered there is same value exclude from at the position Best Case and Worst Case, with *counter* = *k*. Here detail the steps.

$$C(n) = \sum_{k=1}^{C_{worst}(n)} \left(counter \frac{1}{C_{worst}(n)}\right)$$

$$= \sum_{k=1}^{\frac{n(n-1)}{2}} \left(k \frac{1}{\frac{n(n-1)}{2}}\right) = \sum_{k=1}^{\frac{n(n-1)}{2}} \left(k \frac{2}{n(n-1)}\right) = \frac{2}{n(n-1)} \sum_{k=1}^{\frac{n(n-1)}{2}} k$$

$$= \frac{2}{n(n-1)} \left(\frac{\frac{n(n-1)}{2}}{2}\left(1 + \left(\frac{n(n-1)}{2}\right)\right)\right) = \frac{2}{n(n-1)} \left(\frac{n(n-1)}{4}\left(1 + \left(\frac{n(n-1)}{2}\right)\right)\right)$$

$$= \frac{1}{2}\left(1 + \left(\frac{n(n-1)}{2}\right)\right) = \frac{1}{2} + \left(\frac{n^2-n}{4}\right) = \frac{2}{4} + \left(\frac{n^2-n}{4}\right) = \frac{1}{4}\left(n^2 - n + 2\right) \quad (17)$$

from equation (16) and (17) it turns out obtained the same value, so it can be shown that,

$$C(n) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \left(\left((j-i) + \left(\frac{(i-1)}{2}\left((n-1) + (n-i) + 1\right)\right)\right) \frac{1}{C_{worst}(n)}\right)$$

$$= \sum_{k=1}^{C_{worst}(n)} \left(counter \frac{1}{C_{worst}(n)}\right) \quad (18)$$

Based on the equation (18), can be created the general equation that in order to calculate the Average Case, no matter how many layers of sigma notation, then this can create the formulations to solve it with just a single sigma, namely in equation (19) as follows

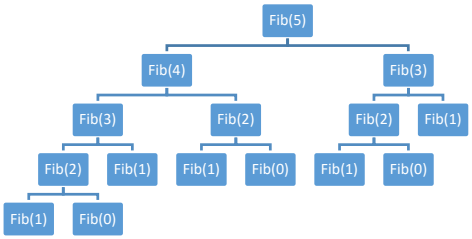
$$C_{average}(n) = \sum_{k=1}^{C_{worst}(n)} \left(k \frac{1}{C_{worst}(n)}\right)$$

$$= \frac{1}{C_{worst}(n)} \sum_{k=1}^{C_{worst}(n)} (k) = \frac{1}{C_{worst}(n)} \left(\frac{C_{worst}(n)}{2} \left(1 + C_{worst}(n)\right) \right)$$

$$= \frac{1}{2} \left(1 + C_{worst}(n)\right) \quad (19)$$

Then, in Fibonacci algorithms, this can be created recursively that aims to present the numbers to a specific sequence. Forms recursive functions, of Fibonacci can be seen as follows Section 3.1 and 3.2.

3.1 Recursive Trees Fib(5)



3.2 Compute Time Complexity

$C(n) = C(n\text{-}1) + C(n\text{-}2) + a$! assumed that $a = 1$, $C(1)=1$ and $C(0)=1$.

$$C(n) = C(n-1) + C(n-2) + 1 \quad (20)$$

a. Outlining equation (20) with single side, and choose $C(n-1)$ as outlining priority, because it has the deepest levels of recursive than $C(n-2)$.

$$= (C(n-2) + C(n-3) + 1) + C(n-2) + 1$$
$$= 2C(n-2) + C(n-3) + 2 \quad (21)$$
$$= (C(n-3) + C(n-4) + 1) + C(n-3) + 2$$
$$= 3C(n-3) + 2C(n-4) + 4 \quad (22)$$
$$= (C(n-4) + C(n-5) + 1) + 2C(n-4) + 4$$
$$= 5C(n-4) + 3C(n-5) + 7 \quad (23)$$
$$= (5C(n-5) + C(n-6) + 1) + 3C(n-5) + 7$$
$$= 8C(n-5) + 5C(n-6) + 12 \quad (24)$$
$$= (8C(n-6) + C(n-7) + 1) + 5C(n-6) + 12$$
$$= 13C(n-6) + 8C(n-7) + 20 \quad (25)$$

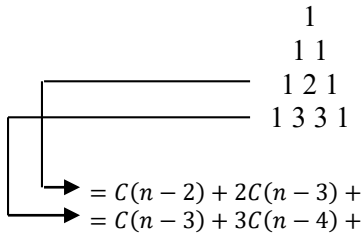
b. Outlining equation (20) with both sides, namely $C(n-1)$ and $C(n-2)$ given same priority.

$$= (C(n-2) + C(n-3) + 1) + (C(n-3) + C(n-4) + 1) + 1$$
$$= C(n-2) + 2C(n-3) + C(n-4) + 3 \quad (26)$$

If formed to $C(n-2)$ and $C(n-3)$ with $C(n-4) = C(n-2) - C(n-3) - 1$, then substituted into the equation (26) will be obtained:

$$= C(n-2) + 2C(n-3) + C(n-4) + 3$$
$$= C(n-2) + 2C(n-3) + (C(n-2) - C(n-3) - 1) + 3$$
$$= 2C(n-2) + C(n-3) + 2 \quad (27)$$

and so on, because of the outlining of the both sides more complex, then selected the outlining of only with a single side that more simple but still the end result is the same. It can be seen that equation (27) produced from the outlining of the both sides is the have same result as the equation (21) that produced from the outlining of a single side. And Pascal triangle pattern is formed from the outlining by the both sides, to support these results, can also be found in previous research [12][13].



If $C(n-3) + 3C(n-4) + 3C(n-5) + C(n-6) + 7$ formed into $C(n-3)$ and $C(n-4)$ with $C(n-6) = C(n-4) - C(n-5) - 1$ and by substituting into $3C(n-5) + C(n-6)$ will be obtained:

$$3C(n-5) + C(n-6) = 3C(n-5) + C(n-4) - C(n-5) - 1$$
$$= C(n-4) + 2C(n-5) - 1$$

then substitute $C(n-5) = C(n-3) - C(n-4) - 1$ into $C(n-4) + 2C(n-5) - 1$ will be obtained:

$$C(n-4) + 2C(n-5) - 1 = C(n-4) + 2C(C(n-3) - C(n-4) - 1) - 1$$
$$= 2C(n-3) - C(n-4) - 3$$

thus obtained:

$$3C(n-5) + C(n-6) = 2C(n-3) - C(n-4) - 3$$

then substitute into the equation $(n-3) + 3C(n-4) + 3C(n-5) + C(n-6) + 7$ thus obtained result:

$$= C(n-3) + 3C(n-4) + 3C(n-5) + C(n-6) + 7$$
$$= C(n-3) + 3C(n-4) + 2C(n-3) - C(n-4) - 3 + 7$$
$$= 3C(n-3) + 2C(n-4) + 4 \quad (28)$$

It can be seen that equation (28) produced from the outlining of the both sides is the have same result as the equation (22) that produced from the outlining of a single side. Binet's formula for the *n*-th Fibonacci number:

$$U_n = \frac{(1+\sqrt{5})^n - (1-\sqrt{5})^n}{2^n\sqrt{5}} \quad (29)$$

If: $n = 0,\ U_n = 0$
$n = 1,\ U_n = 1$
$n = 2,\ U_n = 1$
$n = 3,\ U_n = 2$
$n = 4,\ U_n = 3$
$n = 5,\ U_n = 5$

$n = 6,\ U_n = 8$
$n = 7,\ U_n = 13$
$n = 8,\ U_n = 21$
And so on.

$$= XC\ (n-i) + YC\big(n-(i+1)\big) + (X + (Y-1)) \quad (30)$$

Based on outlining of the equation (21) to (25), and after checked pattern of it, X and Y identified as Fibonacci series, and eg assumed $(n-i) = 1$ and $i = n-1$ has entered the Best Case condition (the last outlining), then

$= XC\ (n-i) + YC(n-i-1) + (X + Y - 1)$
$= XC\ (1) + YC(1-1) + (X + Y - 1)$
because $C(1) = 1$ and $C(0) = 1$ then
$= XC\ (1) + YC(0) + (X + Y - 1)$
$= X(1) + Y(1) + (X + Y - 1)$
$= X + Y + X + Y - 1$
$= 2(X + Y) - 1$

$X = \frac{(1+\sqrt{5})^{i+1} - (1-\sqrt{5})^{i+1}}{2^{i+1}\sqrt{5}}$ and

$Y = \frac{(1+\sqrt{5})^{i} - (1-\sqrt{5})^{i}}{2^{i}\sqrt{5}}$

substitution X and Y into the equation $2(X + Y) - 1$ so obtained,

$= 2(X + Y) - 1$
$= 2\left(\frac{(1+\sqrt{5})^{i+1} - (1-\sqrt{5})^{i+1}}{2^{i+1}\sqrt{5}} + \frac{(1+\sqrt{5})^{i} - (1-\sqrt{5})^{i}}{2^{i}\sqrt{5}}\right) - 1$
substitution $i = n-1$ so obtined,

$$C(n) = 2\left(\frac{(1+\sqrt{5})^{n-1+1} - (1-\sqrt{5})^{n-1+1}}{2^{n-1+1}\sqrt{5}} + \frac{(1+\sqrt{5})^{n-1} - (1-\sqrt{5})^{n-1}}{2^{n-1}\sqrt{5}}\right) - 1$$

$$C(n) = 2\left(\frac{(1+\sqrt{5})^{n} - (1-\sqrt{5})^{n}}{2^{n}\sqrt{5}} + \frac{(1+\sqrt{5})^{n-1} - (1-\sqrt{5})^{n-1}}{2^{n-1}\sqrt{5}}\right) - 1$$

$$C(n) = \frac{2}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^{n} - \left(\frac{1-\sqrt{5}}{2}\right)^{n} + \left(\frac{1+\sqrt{5}}{2}\right)^{n-1} - \left(\frac{1+\sqrt{5}}{2}\right)^{n-1}\right) \text{-1} \quad (31)$$

$C(n)$ have a complexity class $O\left(\left(\frac{1+\sqrt{5}}{2}\right)^{n}\right) \approx O(1.62^{n}) \approx O(2^{n})$

4 Conclusion

From the final result it can be concluded that the time complexity of Fibonacci recursive algorithm is less than 2^n. The value of $C(n)$ can also be obtained in exact form without

any approximation. Thus, the exact result from the outlining with a single side can be said to be more precise than the approximation method. This approximation method replaces the lower bound $C(n\text{-}2)$ from the formula $C(n) = C(n\text{-}1) + C(n\text{-}2) +1$ with the upper bound $C(n\text{-}1)$ to become $C(n) = 2C(n\text{-}1) +1$). It is intended to simplify the outlining in its calculations, but it ignores the optimal results that would be obtained. The suggestion for future research is to develop Fibonacci algorithm in m stages, namely $C(n) = C(n\text{-}1) + C(n\text{-}2) + C(n\text{-}3) + ... + C(n\text{-}m)$, which can also be solved with a single side or other better methods of the outlining.

References

[1] Charles Burnett. (2016, September). Leonard of Pisa (Fibonacci) and Arabic Arithmetic, [Online], Available: http://www.muslimheritage.com/article/leonard-pisa-fibonacci-and-arabic-arithmetic

[2] Clive N. Menhinick. The Fibonacci Resonance and other new Golden Ratio discoveries. OnPerson International Limited, Poynton, Cheshire (2015). 618 pp.+xiv, ISBN: 978-0-9932166-0-2

[3] Sandeep. (2016, September). Fibonacci numbers in Plants: Design of Leaf, Petals, Branches and Flowers, [Online]. Available: http://ultraxart.com/fibonacci-numbers-in-plants-design-of-leaf-petals-branches-and-flowers/

[4] Md. Akhtaruzzaman, Amir A. Shafie, Geometrical Substantiation of Phi, the Golden Ratio and the Baroque of Nature, Architecture, Design and Engineering, International Journal of Arts (2011); 1(1): 1-22.

[5] Manuel Rubio, Bozena Pajak, Fibonacci numbers using mutual recursion, (2005).

[6] Anany Levitin, Introduction To The Design & Analysis of Algorithms, Addison Wesley, (2003).

[7] Watt D.A., Brown D.F. Java Collections. An Introduction to Abstract Data Types, Data Structures, and Algorithms (2001).

[8] JOC/EFR © (2016, September). François Édouard Anatole Lucas, School of Mathematics and Statistics University of St Andrews, Scotland. (1996). Available: http://www-history.mcs.st-andrews.ac.uk/Biographies/Lucas.html

[9] R. Odendahl, (2016, September). Analysis of Algorithms. Available: http://www.cs.oswego.edu/~odendahl/coursework/csc465/notes/04-analysis.html

[10] Alexey Stakhov, The Mathematics of Harmony: From Euclid to Contemporary Mathematics and Computer Science. World Scientific Publishing Co. Pte. Ltd. (2009).

[11] Alexey Stakhov, Samuil Aranson. The “Golden” Non-Euclidean Geometry: Hilbert's Fourth Problem, “Golden” Dynamical System, and the Fine-Structure Constant. (2016).

[12] J. L. Holloway. Algorithms for Computing Fibonacci Numbers Quickly. (1988).

[13] Thomas Koshy. Fibonacci, Lucas, and Pell Numbers, and Pascal’s Triangle. (2011).

[14] Jeffrey J. McConnell. Analysis of Algorithms: An Active Learning Approach, by Jones and Bartlett Publishers, Inc., (2001).