ADA-F23-A1 (Fibonacci Series Time Complexity Analysis)

Muhamamd Bilal

November 3, 2023

Question

Write a C++ program to generate the first n numbers of the Fibonacci series using both iterative and recursive approaches. Analyze both approaches and estimate the time complexity in terms of asymptotic notations. Calculate the execution time taken for both approaches for different values of n. Create a comprehensive report that includes your analysis in asymptotic notation and execution time taken for both approaches. The comparison should encompass text, tables, and graphs.

1 Algorithms

1.1 Iterative

```
void FibIterative(int n)
{
    int a = 0, b = 1;
    if (n <= 0)
    {
        return;
    }
    for (int i = 1; i <= n; i++)
    {
        cout << a << " ";
        int c = a + b;
        a = b;
        b = c;
    }
}
```

1.2 Recursive

```
int FibRecursive(int n)
{
    if (n <= 1)
    {
        return ;
    }
    else
    {
        return FibRecursive(n - 1)
        + FibRecursive(n - 2);
    }
}
```

2 Time Complexity (Asymtotic Notations)

2.1 Iterative Approach

```
void FibIterative(int n)
{
    int a = 0, b = 1;     ........ 1
    if (n <= 0)     ........ 1
    {
        return;
    }
    for (int i = 1; i <= n; ++i)     ........ n+1
    {
        cout << a << " ";     ........ n
        int c = a + b;     ........ n
        a = b;     ........ n
        b = c;     ........ n
    }
}
```

We can express the time complexity of iterative approach as follows:

$$T(n) = 5n + 3 \tag{1}$$

2.1.1 Upper Bound

f(n) = 5n + 3 , g(n) = n , c = 8

$$5n + 3 <= 8(n) \tag{2}$$

2.1.2 Lower Bound

f(n) = 5n + 3 , g(n) = n , c = 1

$$5n + 3 >= n \tag{3}$$

2.2 Recursive Approach

```
int FibRecursive(int n)
{
    if (n <= 1)     ........ 1
    {
        return ;
    }
    else
    {
        return FibRecursive(n - 1)      ........ T(n-1) + T(n-2)
        + FibRecursive(n - 2);
    }
}
```

Now the recurrence equation is written as :

$$T(n) = T(n-1) + T(n-2) + 1 \tag{4}$$

2.2.1 Lower Bound

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

For lower bound, we can assume:

$$T(n-1) = T(n-2)$$
$$T(n) = 2T(n-2) + 1 \quad \text{(i)}$$
$$T(n-2) = 2T(n-4) + 1$$

Putting in (i):

$$T(n) = 2[2T(n-4) + 1] + 1$$
$$T(n) = 4T(n-4) + 2 + 1$$
$$T(n-4) = 4T(n-4) + 2 + 1$$
$$T(n-4) = 2T(n-6) + 1$$

Putting in (i):

$$T(n) = 4[2T(n-6) + 1] + 2 + 1$$
$$T(n) = 8T(n-6) + 4 + 2 + 1$$

Assuming k steps:

$$T(n) = 2^k T(n-2k) + 2^{k-1} + \ldots + 2 + 1$$

Assume $n - 2k = 0$ (i.e., $n = 2k$)

$$k = \frac{n}{2}$$

Let $x = 2^{n/2} + 2^{(n/2-1)} + \ldots + 2 + 1 \quad (1)$

$$-2x = 2^{(n/2+1)} + 2^{(n/2)} + \ldots + 2 \quad (2)$$

Subtracting (2) from (1):

$$x = 2^{(n/2+1)} - 1$$

$$T(n) = O(2^n) \tag{5}$$

2.2.2 Upper Bound

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

For upper bound, we can assume:

$$T(n-2) = T(n-1)$$
$$T(n) = 2T(n-1) + 1 \quad \text{(i)}$$
$$T(n-1) = 2T(n-2) + 1$$

Putting in (i):

$$T(n) = 2[2T(n-2) + 1] + 1$$
$$T(n) = 4T(n-2) + 2 + 1$$
$$T(n-2) = 4T(n-4) + 2 + 1$$
$$T(n-4) = 2T(n-6) + 1$$

Putting in (i):

$$T(n) = 4[2T(n-6) + 1] + 2 + 1$$
$$T(n) = 8T(n-6) + 4 + 2 + 1$$

Assuming k steps:

$$T(n) = 2^k T(n-6k) + 2^{k-1} + \ldots + 2 + 1$$

Assume $n - 6k = 0$ (i.e., $n = 6k$)

$$k = \frac{n}{6}$$

Let $x = 2^{n/6} + 2^{(n/6-1)} + \ldots + 2 + 1 \quad (1)$

$$-2x = 2^{(n/6+1)} + 2^{(n/6)} + \ldots + 2 \quad (2)$$

Subtracting (2) from (1):

$$x = 2^{(n/6+1)} - 1$$

$$T(n) = O(2^n) \tag{6}$$

3 Comparison of Execution Times

In this comparison, we measured the execution times in milliseconds for both iterative and recursive approaches when calculating the Fibonacci sequence for various values of n. The purpose of this analysis is to understand the performance differences between these two methods.

n	Iterative (ms)	Recursive (ms)
4	2.002	0
8	9.376	0
12	30.000	0
16	15.580	0
20	20.000	0
24	31.517	4
28	61.916	35
32	30.002	314
36	40.046	2318

3.1 Execution Time and Values of n Comparison

3.2 Analysis

- For small values of n (e.g., 4, 8, 12, 16), the iterative approach performs significantly faster than the recursive approach. The recursive approach in these cases has very low execution times (close to 0), which is expected, as it has relatively simple calculations.
- As n increases, the recursive approach's execution time starts to grow significantly, reaching milliseconds for larger values. This indicates that the recursive method suffers from exponential time complexity.
- The iterative approach also experiences an increase in execution time with larger n, but the growth is relatively slower compared to the recursive approach. It maintains a more linear time complexity.
- For $n = 24$, the recursive approach takes 4 milliseconds, indicating that the algorithm is starting to exhibit a noticeable time delay.
- By the time $n = 36$, the recursive approach has a substantial execution time of 2318 milliseconds, demonstrating that it's considerably slower for larger values of n.

In summary, the iterative approach is more efficient and suitable for calculating the Fibonacci sequence, especially for larger values of n, where the recursive approach suffers from severe performance issues due to its exponential time complexity.

3.3 Graphs

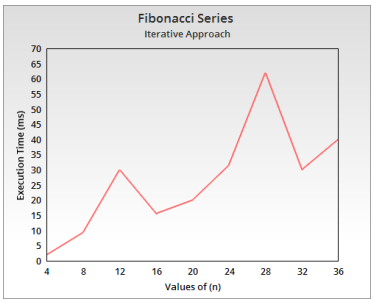

Figure 1: Iterative Approach

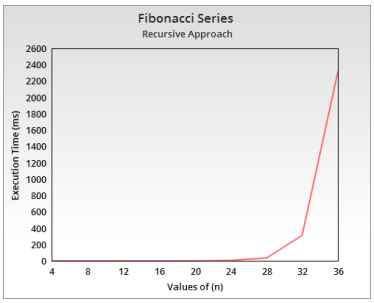

Figure 2: Recursive Approach

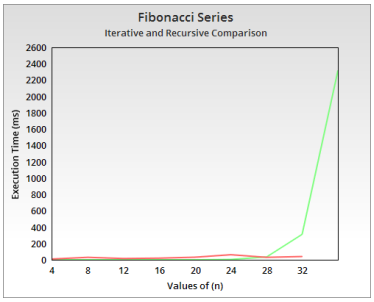

Figure 3: Comparison