Understand Time Complexity of Recursion

Yumo Bai , Sayeed Ghani

September 2022

1 Introduction

In lecture 5 & 6, we discussed the time and space complexity of a few recursion algorithms. Because of the nature of recursion, it may be somewhat difficult to get the result by intuition. This document will provide a visual representation for Fibonacci Sequence recursion program to help you better understand why it is $O(2^n)$

2 Understanding Fibonacci Recursion

Fibonacci sequence is such that each number is the sum of the two preceding ones. The sequence starts from F(2), with F(0)=0 and F(1)=1. Hence the following are first few terms of the sequence:

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...$$

Based on the nature of this sequence, we could easily find a mathematical formula to represent this sequence as:

$$F_0 = 0, F_1 = 1 \qquad (1)$$

$$F_n = F(n-1) + F(n-2) \quad n \geq 2 \quad (2)$$

Therefore, from this mathematical formula, we could write a recursive method for calculating nth term of Fibonacci sequence:

```
public int fib(int n)
{
        if(n==1||n==2)
        {
                return 1;
        }
        return fib(n-1)+fib(n-2);
}
```

3 Analysis of Fibonacci Sequence Program

Consider $fib(n), n \geq 2$, from the definition above we see that $fib(n) = fib(n-1) + fib(n-2)$. The following graph provides a visual representation of how fib(n) is calculated by the recursion method.

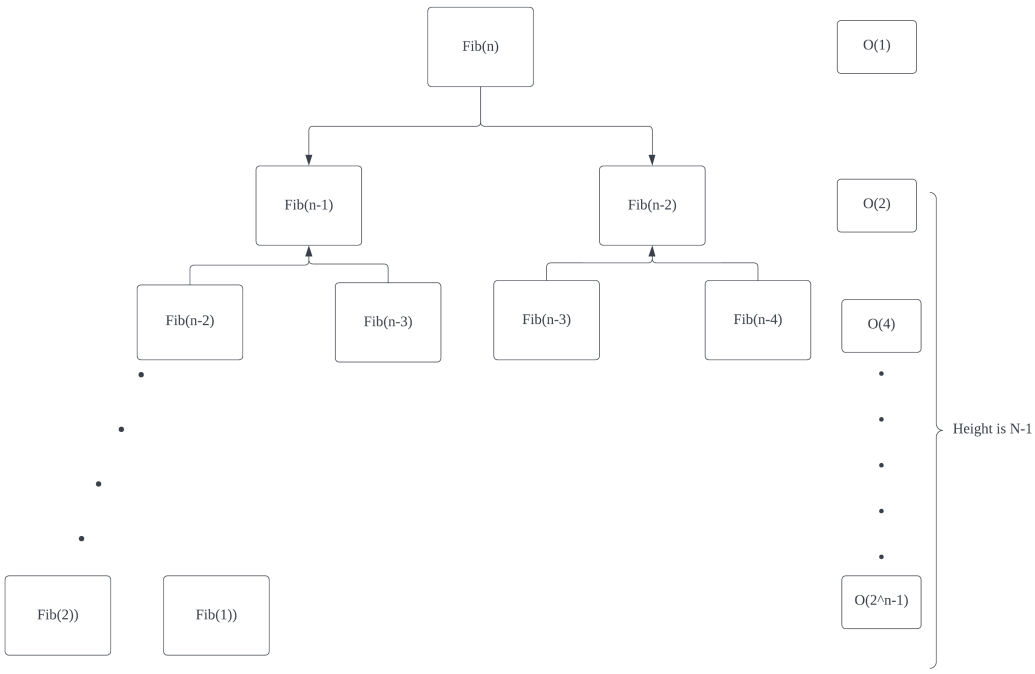

Figure 1: Visual Representation of Recursion Program

Based on the observation of graph above, we could calculate the time complexity on the basis of the level. The total time complexity would be the sum of all the levels of the diagram.

$$1 + 2 + 4 + ... + 2^{n-1} = \sum_{i=0}^{n-1} \quad (2^n)$$

It is easily to find that the above series is geometric series that the sum of it could be calculated as:

$$S_n = \frac{a_1(1 - r^n)}{1 - r}$$

Applying the formula we have:

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

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

Therefore, we can see that the time complexity of this recursion approach is $O(2^n)$

4 Extension

Try to understand why the time complexity of program for factorial is $O(n)$

```
public int factorial(int n)
{
        if(n==1)
        {
                return 1;
        }
        return n*factorial(n-1);
}
```

5 Challenge Questions

Construct your own recursion algorithm for the following question:

1.Find the 2 to the nth power (e.g: $2^{10} = 1024$)

2.Bob can walk through one step or two steps at a time. Calculate how many combinations he could have if there are total n steps.

3.Find the 2 to the nth power with O(log(n)) complexity

If you solved the problem or have questions, feel free to discuss with LA/TA during office Hour.