CPSC 319 Assignment 1: Analysis of Algorithm

Name:Shanzi Ye

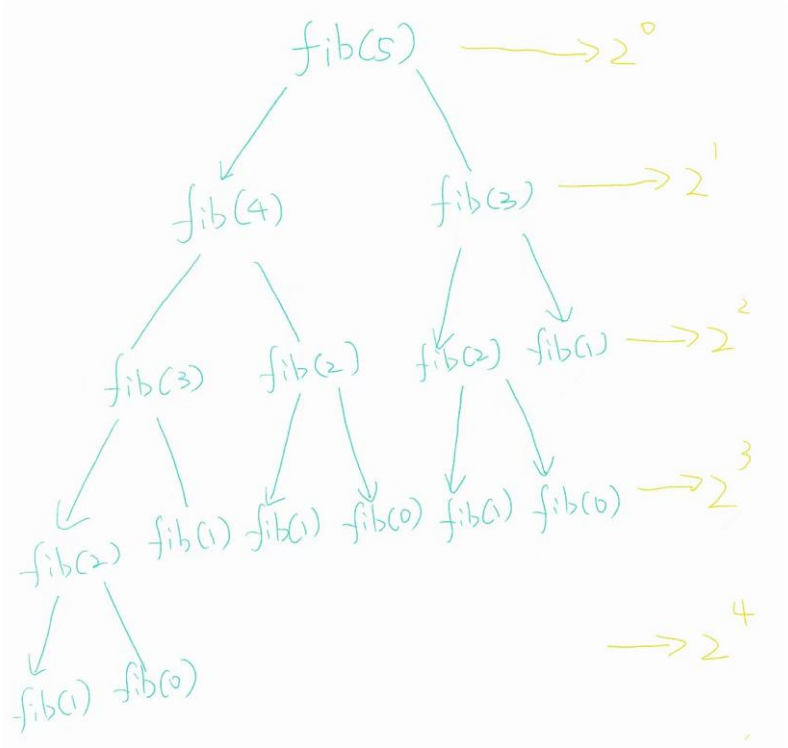
Instructor:Bajwa, Sohaib

Submission Date: July,10,2022

1. Time complexity analysis for recursion algorithm:

Its time complexity is O(2^n).After several experiments, I notice that if we input a integer larger than 45, the computer will take much longer time to compute. This is because running a recursive algorithm requires a larger number of function calls. If the user inputs a large number, the computer consumes a lot of time and resources to push the stack again and again, thereby reducing the efficiency of the program. Undoubtedly its efficiency is much lower than the iterative and memorization algorithm.

Firstly let' s assume n=5;



From the above binary tree, we can see that additional two nodes will be generated when each node branches off(except 0 and 1),which makes the operations grow exponentially. If n equals to 5, the program will take approximately (2^4-1) operations(nodes) to compute. Similarly, if the user inputs n, the total operations to compute Fibonacci will be approximately 2^(n-1). It is obvious that our binary tree is always asymmetric no matter what n is, hence the actual number of operation is not exactly to 2^(n-1), but close to. However, when determining the time complexity, we tend to estimate the worst case scenario. Therefore,we can simply ignore the constant and conclude that the time complexity of the recursive Fibonacci is approximately O(2^n). In my opinion, this algorithm is not recommended.

2.Time complexity analysis for iterative algorithm:

We can simply follow the steps we discussed in class. We just need to find the growth rate function and determine its Big-O notation.

Line#	Algorithm	Cost	Times	Comments
3	long firstNum = 0;	C1=1	1	one op: assign
4	long secondNum = 1;	C2=1	1	one op: assign
5	long thirdNum = 0;	C3=1	1	one op: assign
6	if (n == 0)	C4=1	1	One op: comparing two numbers
8	return 0;	C5=1	1	one op:return from a method
10	else if (n == 1)	C6=1	1	One op: comparing two numbers
12	return 1;	C7=1	1	one op:return from a method
		C8=1	1	Loop initializing (assigning a value)
16	for (int i = 2; i <= n ; i++) {	c9=1	n	Loop termination test: • a comparison i < n each time • n-1 successes and one failure
		C10=2	n-1	Loop incrementation: • Two ops: an addition and an assignment • done n-1 times
17	thirdNum = firstNum + secondNum;	C11=2	n-1	Two ops per iteration (i.e.add,assign). • executed n-1 times
18	firstNum = secondNum;	C12=1	n-1	one ops per iteration (i.e.assign). • executed n-1 times
19	secondNum = thirdNum;	C13=1	n-1	one ops per iteration (i.e.assign). • executed n-1 times
22	return thirdNum;	C14=1	1	one op:return from a method

$$
\begin{aligned}
F(n)&=C1*1+C2*1+C3*1+C4*1+C5*1+C6*1+C7*1+C8*1+C9*n+C10*(n-1)+C11*(n-1)+C12*(n-1)\\
&+C13*(n-1)+C14*1\\
&=C1+C2+C3+C4+C5+C6+C7+C8+C9*n+C10*n-C10+C11*n-C11+C12*n-C12+C13*n-C13+C14\\
&=(C9+C10+C11+C12+C13)*n+(C1+C2+C3+C4+C5+C6+C7+C8-C10-C11-C12-C13+14)\\
&=(1+2+2++1+1)*n+(1+1+1+1+1+1+1+1-2-2--1-1+1)\\
&=7n+3
\end{aligned}
$$

Therefore,the time complexity of algorithm3 is O(n).Compared with Algorithm 1, we can know that the iterative algorithm is much more efficient.

3.Time complexity analysis for memorization algorithm:

The time complexity of memorization algorithm is O(n). By adding a layer of cache to store previously calculated values, the program first searches the cache when a new function needs to be calculated. If the result already exists in the cache, the program will directly return the memorized result from the cache instead of calculating it for the second time. Unlike algorithm 1, memorization algorithm will be called only once for each value from 0 to n. Therefore, its time complexity is approximately O(n). Even though both algorithm 1 and algorithm 2 are based on recursive algorithm, algorithm 2 is much more efficient.

Conclusion:

After our analysis, we know that the time complexity for both algorithm 2 and 3 are O(n). If we compare the scatterplot for both algorithm, we find that with n increases, iterative algorithm takes less time to executive. Thus, algorithm 3 is slightly better than algorithm 2 and both algorithm 2 and 3 are much better than algorithm 1.

From best to worse:

Iterative algorithm > memorization algorithm > recursive algorithm