Heaven's Light is Our Guide

Rajshahi University of Engineering and Technology
Department of Computer Science and Engineering

Course No: CSE.2202
Course Title: Sessional based on CSE.2201 (Computer Algorithms)

Lab Report No: 01
Lab Report On: Complexity of Algorithms (Fibonacci Series)

Submitted By
Md. Ariful Islam
Roll No: 1803046
Section: A
Department: CSE

Submitted To
Dr. Md. Ali Hossain
Associate Professor
Dept. of CSE,RUET

Date: 06-02-2021

❖ **Problem Statement:** The problem is to generate a recursive algorithm for creating the series of Fibonacci numbers up to a given number of elements. The number of elements should be read from a file **fibonacci.txt**. Then to implement the code of the algorithm, finding out the total number of steps. Finally to compare the number of steps (i.e., at least 10 comparisons) with an algorithm which do not use recursive method for generating the Fibonacci numbers (Graphs should be included if required).

❖ **Details description and Algorithm:** The process which repeat itself in a self-similar way is called Recursion. The function which calls itself directly or indirectly is called Recursive Function. The Fibonacci numbers are the integer sequence like below:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55,……

And can be defined by a recurrence relation

$$F_n = F_{n-1} + F_{n-2}$$

Where $F_0 = 0$ and $F_1 = 1$.

Let us consider the following algorithm for **Recursive Function**:

Fib (a) [Fib is a function that's parameter "**a"** is an integer **]**

1. **IF a <= 1** then**:**
   Set **FB[a]:= a .** **[**FB is a globally declared integer array initialized as -1**]**
   [End of **IF** structure **]**
2. **IF FB[a] != -1** then:
   RETURN FB[a] . **[**It means the value of **FB[a]** is already calculated**]**
   [End of **IF** structure **]**
3. Set **FB[a]:= Fib(a-1) + Fib(a-2) .** **[**Stores recursive value in **FB[a]**]
4. **RETURN FB[a] .** **[** Returns the stored value in **FB[a]]**
5. Exit.

And the algorithm for first **a** Fibonacci number in a **Non-recursive** way:

- **NonR(a)**
  1. Set **Y:= 0** and **Z:= 1** **[** Initialize 1st and 2nd value **]**
  2. Print **Y** and **Z**
  3. Repeat **i = 2** to **a-1** by **1**
  4. Print **Y+Z .** **[** Next value of sequence **]**
  5. Set **TEMP:= Z , Y:= Z , Z:= Y + TEMP .** **[** Calculate next values **]**

     [End of Repeat **step 3** loop **]**
  6. Exit.

❖ Implemented Code:

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;

ll Count = 0;
ll fb[101];

ll fib(int a){
   if(a<=1){
      fb[a]=a;
      Count+=3;
      return a;
   }
   if(fb[a]!=-1){
      Count+=2;
      return fb[a];
   }
   fb[a]=fib(a-1)+fib(a-2);
   Count+=2;
   return fb[a];
}


int main(){
   int a;
   string x;
   ifstream f1,f2;
```

```
    f1.open("fibonacci.txt");

    while(!f1.eof()){
        memset(fb,-1,sizeof(fb));
        Count = 5;
        f1>>x;
        a=stod(x); Count+=2;
        fib(a);
        Count+=1;
        for(int i=0;i<a;i++){
            Count+=3;
            cout<<fb[i]<<" ";
        }
        cout<<"\n"<<endl;
        Count+=2;
        cout<<"N : "<<a<<"  Count: "<<Count<<"\n"<<endl;
    }

    f2.open("fibonacci.txt");

    while(!f2.eof()){
        Count=5;
        f2>>x;
        ll y=0,z=1;
        a=stod(x); Count+=4;
        cout<<"\n"<<y<<" "<<z<<" ";
        for(int i=0;i<a-2;i++){
            cout<<y+z<<" "; Count+=6;
            ll temp=y;
            y=z;
            z+=temp;
        }
        Count+=2;
        cout<<"\n\nN : "<<a<<"  Count: "<<Count<<"\n"<<endl;

    }

    return 0;
}
```

❖ **Sample Input and Output:**

Input: fibonacci.txt

```
5
10
15
20
25
30
35
40
45
50
55
60
65
70
```

Output:

```
0 1 1 2 3
N : 5  Count: 46

0 1 1 2 3 5 8 13 21 34
N : 10  Count: 81

0 1 1 2 3 5 8 13 21 34 55 89 144 233 377
N : 15  Count: 116

0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181
N : 20  Count: 151

0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368
N : 25  Count: 186

0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368
75025 121393 196418 317811 514229
N : 30  Count: 221
```

0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040 1346269 2178309 3524578 5702887
N : 35 Count: 256

0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040 1346269 2178309 3524578 5702887 9227465 14930352 24157817 39088169 63245986
N : 40 Count: 291

0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040 1346269 2178309 3524578 5702887 9227465 14930352 24157817 39088169 63245986 102334155 165580141 267914296 433494437 701408733
N : 45 Count: 326

0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040 1346269 2178309 3524578 5702887 9227465 14930352 24157817 39088169 63245986 102334155 165580141 267914296 433494437 701408733 1134903170 1836311903 2971215073 4807526976 7778742049
N : 50 Count: 361

0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040 1346269 2178309 3524578 5702887 9227465 14930352 24157817 39088169 63245986 102334155 165580141 267914296 433494437 701408733 1134903170 1836311903 2971215073 4807526976 7778742049 12586269025 20365011074 32951280099 53316291173 86267571272
N : 55 Count: 396

0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040 1346269 2178309 3524578 5702887 9227465 14930352 24157817 39088169 63245986 102334155 165580141 267914296 433494437 701408733 1134903170 1836311903 2971215073 4807526976 7778742049 12586269025 20365011074 32951280099 53316291173 86267571272 139583862445 225851433717 365435296162 591286729879 956722026041
N : 60 Count: 431

0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040 1346269 2178309 3524578 5702887 9227465 14930352 24157817 39088169 63245986 102334155 165580141 267914296 433494437 701408733 1134903170 1836311903 2971215073 4807526976 7778742049 12586269025 20365011074 32951280099 53316291173 86267571272 139583862445 225851433717 365435296162 591286729879 956722026041 1548008755920 2504730781961 4052739537881 6557470319842 10610209857723

N : 65 Count: 466

0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040 1346269 2178309 3524578 5702887 9227465 14930352 24157817 39088169 63245986 102334155 165580141 267914296 433494437 701408733 1134903170 1836311903 2971215073 4807526976 7778742049 12586269025 20365011074 32951280099 53316291173 86267571272 139583862445 225851433717 365435296162 591286729879 956722026041 1548008755920 2504730781961 4052739537881 6557470319842 10610209857723 17167680177565 27777890035288 44945570212853 72723460248141 117669030460994

N : 70 Count: 501

0 1 1 2 3

N : 5 Count: 29

0 1 1 2 3 5 8 13 21 34

N : 10 Count: 59

0 1 1 2 3 5 8 13 21 34 55 89 144 233 377

N : 15 Count: 89

0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181

N : 20 Count: 119

0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368

N : 25 Count: 149

0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229

N : 30 Count: 179

0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040 1346269 2178309 3524578 5702887

N : 35 Count: 209

0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040 1346269 2178309 3524578 5702887 9227465 14930352 24157817 39088169 63245986

N : 40 Count: 239

0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040 1346269 2178309 3524578 5702887 9227465 14930352 24157817 39088169 63245986 102334155 165580141 267914296 433494437 701408733

N : 45 Count: 269

0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040 1346269 2178309 3524578 5702887 9227465 14930352 24157817 39088169 63245986 102334155 165580141 267914296 433494437 701408733 1134903170 1836311903 2971215073 4807526976 7778742049

N : 50 Count: 299

0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040 1346269 2178309 3524578 5702887 9227465 14930352 24157817 39088169 63245986 102334155 165580141 267914296 433494437 701408733 1134903170 1836311903 2971215073 4807526976 7778742049 12586269025 20365011074 32951280099 53316291173 86267571272

N : 55 Count: 329

0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040 1346269 2178309 3524578 5702887 9227465 14930352 24157817 39088169 63245986 102334155 165580141 267914296 433494437 701408733 1134903170 1836311903 2971215073 4807526976 7778742049 12586269025 20365011074 32951280099 53316291173 86267571272 139583862445 225851433717 365435296162 591286729879 956722026041

N : 60 Count: 359

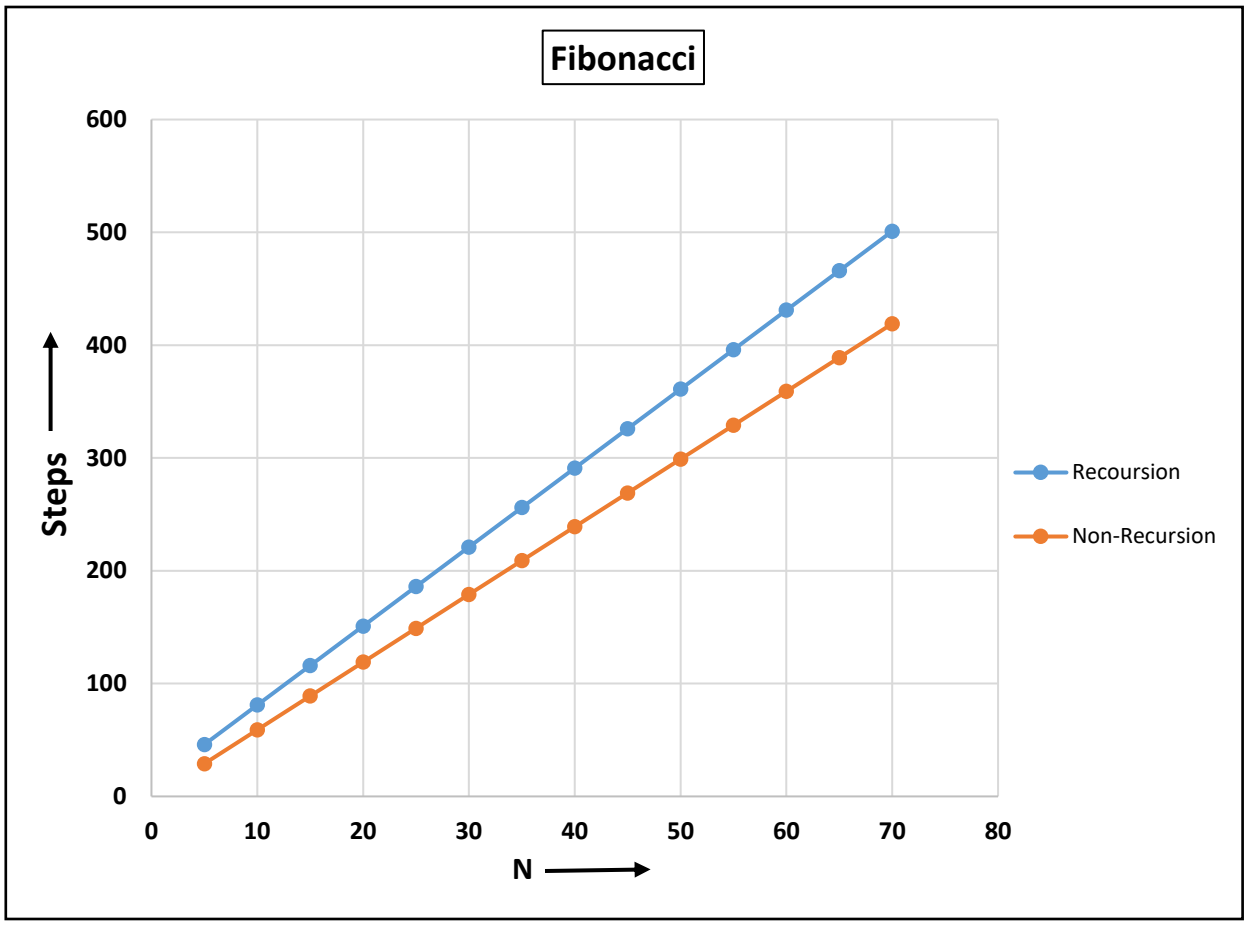
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040 1346269 2178309 3524578 5702887 9227465 14930352 24157817 39088169 63245986 102334155 165580141 267914296 433494437 701408733 1134903170 1836311903 2971215073 4807526976 7778742049 12586269025 20365011074 32951280099 53316291173 86267571272 139583862445 225851433717 365435296162 591286729879 956722026041 1548008755920 2504730781961 4052739537881 6557470319842 10610209857723

N : 65 Count: 389

0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040 1346269 2178309 3524578 5702887 9227465 14930352 24157817 39088169 63245986 102334155 165580141 267914296 433494437 701408733 1134903170 1836311903 2971215073 4807526976 7778742049 12586269025 20365011074 32951280099 53316291173 86267571272 139583862445 225851433717 365435296162 591286729879 956722026041 1548008755920 2504730781961 4052739537881 6557470319842 10610209857723 17167680177565 27777890035288 44945570212853 72723460248141 117669030460994

N : 70 Count: 419

Graph:



❖ **Discussion & Conclusion:** As recursion is a backtracking process, the steps are higher than the non-recursive process. We can see in the graph that the line of recursion is more vertical than the non-recursion line.

So we can say that the non-recursive process is more time efficient than the recursive process.

 # END #