Design And Analysis Of Algorithms - Assignment 1
Group - 2

Aditya Raj(IIT2017005) Mayank Mrinal(IIT2017006)
Mohit Pahuja(IIT2017007) Arashpreet Singh(IIT2017008)

***Abstract*— This document is a complete design and analysis of algorithm of checking whether a number is fibonacci or not.**

I. INTRODUCTION

The Fibonacci numbers, commonly denoted F_n form a sequence, called the Fibonacci sequence, such that each number is the sum of the two preceding ones, starting from 0 and 1. That is,
$F_0 = 0, \quad F_1 = 1,$
and
$F_n = F_{n-1} + F_{n-2},$
$for \quad n > 1.$
This problem requires us to find whether a given number is fibonacci or not. We have design two algorithms . The first one is a repetitive algorithm where we generate nth fibonacci number by adding previous two . The second one is direct result of a mathematical proof.

This report further contains:
II. Algorithm Design
III. Algorithm Analysis
IV. Experimental Study
V. Conclusion
VI. References

II. ALGORITHM DESIGN

A. First Algorithm

The Fibonacci numbers are the numbers in the following integer sequence.
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ..

In mathematical terms, the sequence F_n of Fibonacci numbers is defined by the recurrence relation

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

with seed values

$$F_0 = 0 \text{ and } F_1 = 1.$$

Any input number n is given . Variable a and b are initialized with 0 and 1 respectively. Now we check n for base case condition, if n is 0 or 1 we return true. We repeat a while loop till c less than n. Now inside while loop check if c is equal to n , return true , else assign b to a, c to b and $a + b$ to c. Now after the while loop ends we return false as c exceeds the required number.

Algorithm 1

```
Input:n
a ← 0
b ← 1
if n == a or n == b then
    return true
end if
c ← a + b
while c ≤ n do
    if c == n then
        return true
    else
        a ← b
        b ← c
        c ← a + b
    end if
end
return false
```

B. Second Algorithm

- **Mathematical Result:** a positive integer n is a Fibonacci number if and only if either $5n^2 + 4$ or $5n^2 - 4$ is a perfect square.
  Proof:
  Let $F_0 = 0$, $F_1 = 0$, $F_{r+1} = F_r + F_{r-1}$ be the Fibonacci series and $L_0 = 2$, $L_1 = 1$, $L_{r+1} = L_r + L_{r-1}$ be the Lucas series. It is well known that

  (1) $$(-1)^r + F_r^2 = F_{r+1} * F_{r-1}$$

  (2) $$L_r = F_{r+1} + F_{r-1}$$

  Subtracting four times the first from the square of the second equation, we have

  $$L_r^2 - 4(-1)^r - 4F_r^2 = (F_{r+1} - F_{r-1})^2 = F_r^2$$

  $$5F_r^2 + 4(-1)^r = L_r^2$$

  Thus if n is a Fibonacci number, either $5n^2 + 4$ or $5n^2 - 4$ is a perfect square.

- We are given n as input . Assign a as $5n^2$+4 now initialise a integer variable s to square root of a. If s^2 is equal to a then return true. Now replace a by a-8

,again assign s as square root of a , check of $s^2 = a$ then return true. If neither of these conditions holds then return false.

Algorithm 2

Input:n
a←$5n^2 + 4$
int(s)← $\sqrt{a}$
if s*s == a **then**
return true
end if
a ← $a - 8$
int(s)← $\sqrt{a}$
if s*s == a **then**
return true
end if
return false

III. ALGORITHM ANALYSIS

A. Time Complexity

1) First Algorithm:

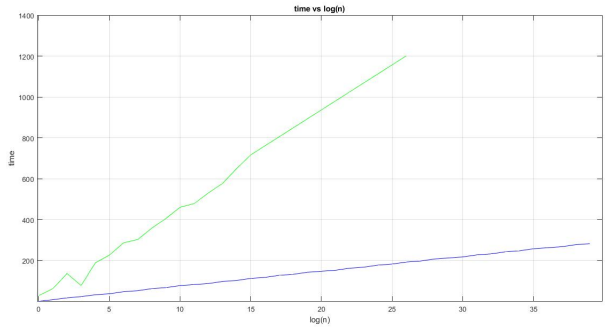
- This algorithm is a repetitive algorithm . We compute the next term of sequence by adding previous two terms.
- Assume the input as n. Let F_k be the Fibonacci number just before or equal to n.
- The time complexity of this algorithm is order of k.
- We found the time complexity for best case , average case and worst case are coming to be same i.e. $O(k)$.

2) Second Algorithm:

- Let the input be n . In this algorithm we check if $5n^2+4$ or $5n^2 - 4$ is perfect square or not.
- If it is a perfect square then the input n will be a fibonacci number.
- To compute the square root of $5n^2 + 4$ or $5n^2 - 4$ we need $O(log(5n^2)) = O(log(n^2))$ i.e. ultimately $O(log(n))$.

B. Space Complexity

- Since the number of memory location is predefined in both the algorithms therefore the space complexity is $O(1)$.



The blue curve is for algorithm 1 and green curve for algorithm 2.

- Since we use long long int for storing a number in algorithm 2 therefore range of number we can store is 10^{18} and as we need to store $5n^2 + 4$ thus the range of n is limited to 10^9 in algorithm 2.
- Since only linear term is used in algorithm 1 therefore the maximum range of input which can be taken is 10^{18}.

IV. EXPERIMENTAL STUDY

In algorithm 1 , the time complexity is coming in order of $log(n)$ i.e. $O(a.log(n))$ where a is any constant.
In algorithm 2 , the time complexity is also coming in order of $log(n)$ i.e. $O(b.log(n))$ where b is other constant.
By the graph we can see that $a > b$.

The integrated Development environment of C is used for processing the algorithm and graphical analysis is done using MATLAB plot function.

V. CONCLUSION

Through this paper we proposed the algorithm to check if a given number is a fibonacci number or not. The first algorithm was an repetitive algorithm and has considerable complexity i.e. order of $O(log(n))$. In second algorithm the time complexity is also in order of $O(log(n))$
but by experimental studies it comes that Algorithm 1 has less time complexity that algorithm 2 because the slope of the curve of algorithm 1 is less than that of algorithm 2 .

VI. REFERENCES

1.https://en.wikipedia.org/wiki/Fibonacci number.
2.https://www.geeksforgeeks.org/program-for-nth-fibonacci-number.
3.[CLR96] Thomas H. Cormen, Charles E. Leiserson, and Ronald L Rivest. Introduction to algorithms. The MIT press, 2nd edition, 1996.
4.[DW96] Nell Dale and Henry M. Walker. Abstract data types: specifications, implementations, and applications. D. C. Heath and Company, Lexington, MA, USA, 1996.
5.https://www.techiedelight.com/find-square-root-using-binary-search-algorithm/