DAA Assignment No. : 1
Locate the smallest Fibonacci number in a randomly generated array of size 1000 where $(10^4 <$ elements of an array $< 10^6)$

Lovely Digra
IIT2017090

Abhishek Vishwakarma
IIT2017091

Anuj Raj
IIT2017092

***Abstract*—Our required aim is to design and analyze an optimal algorithm to find a location of the smallest Fibonacci number in a randomly generated array(arr[]) of size 1000 whose elements lies in a given range $(10^4 < x < 10^6)$. For an ease store all Fibonacci numbers within that range in an array (fib[]). Then iterate through arr[] and search for an element(arr[i]) in fib[] by using binary search. If arr[i] is present in fib[] then return i else print no such number exists in arr[]. While designing our problem we calculated the time complexity for the algorithm and plotted graphs of time complexity vs size of an array. During the analysis, we found out that the algorithm has a time complexity of O(n*log(m)) where n,m are the size of arr[], fib[] respectively.**

I. INTRODUCTION

We were required to design and analyze an algorithm to find a location of the smallest Fibonacci number in a randomly generated array(arr[]) of size 1000 whose elements lies in a given range $(10^4 < x < 10^6)$. As we need to generate an array of some random numbers of given size 1000. For this purpose, we can use rand() function.

Now a question arises what are Fibonacci numbers?

Fibonacci numbers are from a mathematical sequence in which each number is a sum of two preceding number, starting from 0 and 1. In mathematical terms , $F_n = F_n - 1 + F_n - 2$ This sequence can be written like this 0 ,1 ,1 ,2 ,3 ,5 , 8, 13, 21 ..... .

II. ALGORITHM DESIGN AND EXPLANATION

For the problem, we followed the following approach to design an algorithm:

1. Firstly, generate an array(fib[]) consisting of Fibonacci numbers within given range by using Dynamic programming(Memorization).
2. Next our aim is to generate an array(arr[]) consisting of randomly generated positive integers by using rand() function.
3. Now iterate randomly generated array and check whether the element is Fibonacci number or not by using binary search in fib[].
4. Check if element(arr[i]) is present in array(fib[]) storing Fibonacci number if yes then store it's location in variable small otherwise move to step 6.
5. For updating value of small check if it's satisfy the condition $arr[small] < arr[i]$ otherwise let it be the same.
6. Now move to next element in arr[] and repeat step 4.Keep repeating these steps until the value of iterator is less than size of randomly generated array.
7. At last if the initialized value of small is updated then print the value at small index i.e arr[small] otherwise print there is no Fibonacci no. present in arr[] .

The pseudo-code for this algorithm is:-

Given: Size(n) and constraints on element$(10^4, 10^6)$ of an array which we need to generate by using random function.

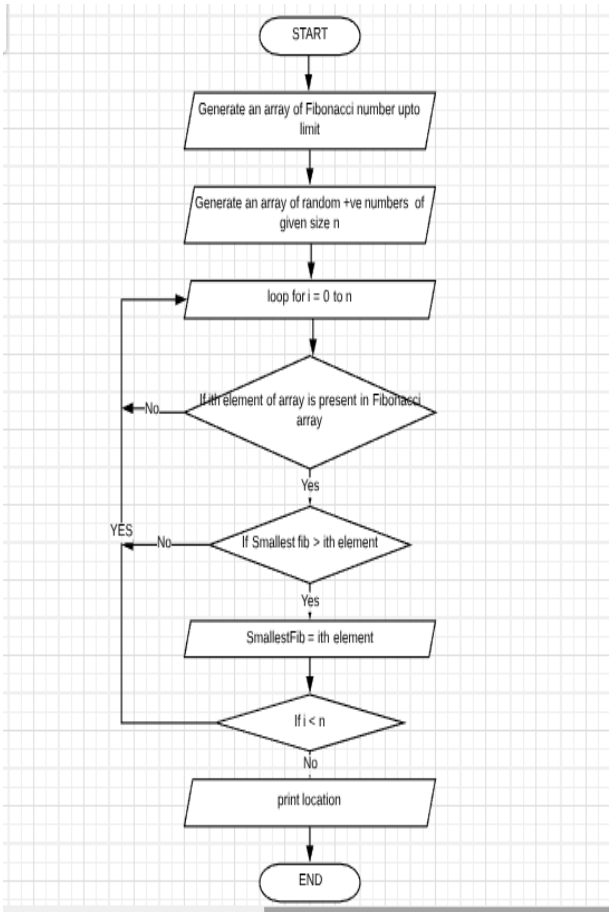

Fig. 1. FlowChart Explanation

Algorithm 1 PseudoCode for the problem

$iterations \leftarrow$ n
$small \leftarrow$ -1
$leftLimit \leftarrow 10^4$
$rightLimit \leftarrow 10^6$
procedure $createFibMap(a0, a1, rightLimit)$
    $fib[0] \leftarrow a0$
    $fib[1] \leftarrow a1$
    for all $i = 1$ to rightLimit **do**
        $fib[i] = fib[i-1] + fib[i-2]$
procedure $randArray(arr, size, leftLimit, rightLimit)$
    for all $i = 1$ to size **do**
        arr[i] = rand() % ((rightLimit-1) - (leftLimit+1) + 1)
+ leftLimit+1
for all $i = 1$ to size **do**
    //Check if arr[i] is present in fib[]
    if $binarySearch(arr[i])$ **then**
        $small \leftarrow i$
if $small! = -1$ **then**
    print small

III. EXPERIMENTAL STUDY

For the experimental analysis of this algorithm, we varied the size of randomly generated array and time taken for an algorithm to terminate.We plotted the graphs for the same.

Best Case : When the smallest Fibonacci number would be at first index.
Worst Case : When the randomly generated array is in decreasing order and each element is Fibonacci number.

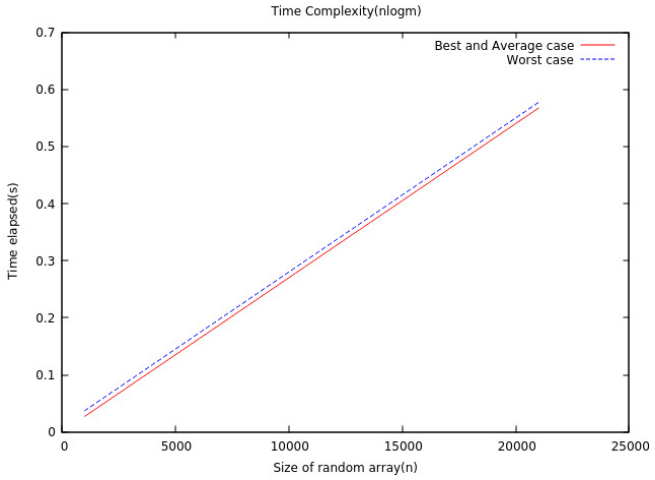

Fig. 2. Graphical Explanation

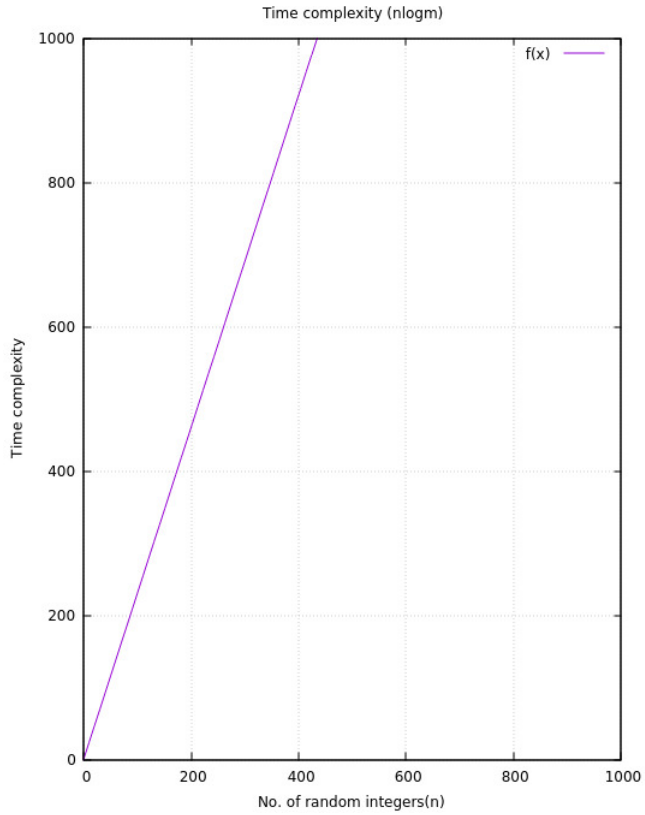

Fig. 3. Graphical Explanation

IV. TIME COMPLEXITY AND DISCUSSION

By analyzing the problem we reached to the point that the time complexity of finding a Fibonacci number in randomly generated array, provided the size of an array and a constraints on element of array would be:
Best Case : omega(log(m))
Worst Case : O(n*log(m)) where n , m are size of Fibonacci and randomly generated array.

V. CONCLUSION

We designed an algorithm to check whether a number is a Fibonacci number or not and whether it lies in an array randomly generated by us or not if yes then print it's index otherwise no such number exists in an array and the same has been stated in this paper. We also analyzed the algorithm and plotted the graphs for time complexity vs size of a randomly generated array.

REFERENCES

[1] $https : //www.youtube.com/watch?v = 9k - l_o l9joklist = PLBA14749045FC203B$

[2] $https : //www.geeksforgeeks.org/program - for - nth - fibonacci - number$

[3] https://www.overleaf.com/learn/latex/Tutorials