SJIF Impact Factor 2022: 8.197| ISI I.F. Value:1.241| Journal DOI: 10.36713/epra2016 ISSN: 2455-7838(Online)
EPRA International Journal of Research and Development (IJRD)
Volume: 7 | Issue: 4 | April 2022 - Peer Reviewed Journal

IMPLEMENTATION OF LUCAS MATHEMATICAL MODEL TO CHECK THE GIVEN NUMBER IS PRIME OR NOT. FURTHER COMPARISON BETWEEN LUCAS METHODOLOGY OF STORING PRIME NUMBERS ON SECONDARY STORAGE DEVICE WITH STORING THE NUMBERS ON PRIMARY STORAGE DEVICE. - A CASE STUDY

6138 – Cadet N Akshayan
Class- XII 2021-22, Sainik School Amaravathinagar
Post: Amaravathinagar,Udumalpet Taluka,Tirupur Dt,Tamilnadu State

ABSTRACT
In computer science efficiency of a program solely depends on time factor of processing statement or statement block. Further the amount of memory it is being used for processing also matters in calculating the space complexity of the program.

All computer languages support sequencing, selection and iteration and file handling methods. The syntax and semantics in a language differs because of construction compiler or an interpreter is different in nature.

This manuscript specifically examines the execution of checking the given number fall under category of prime number or not using Lucas Mathematical Model. A Lucas prime sequence generated and stored on secondary storage device to check the prime number and further comparing with the normal approach of checking a prime number. In addition to this time complexity and space complexity of Lucas prime sequence is examined. The purpose is to provide a alternative methodology for checking the prime numbers.

KEYWORDS: *Lucas Mathematical Model (LMM), Lucas Numbers(ln), Lucas Sequence(ls), Prime Numbers (pn), Fibonacci numbers(fn), Runtime Complexity (rc), Big OO(n), Big Theta Θ(n), Big Omega Ω(n), Generalised approach (ga)*

1. INTRODUCTION

The Lucas numbers (ln) are series of an integer sequence. This methodology was conceptualised by a mathematician Francois Edouard Anatole Lucas. Lucas numbers and Fibonacci numbers are closely related to each other in term of their sequence.

The Lucas sequence(ls) and Fibonacci sequence(fs) have the same function adding the previous two terms but have different starting terms. This produces a sequence where the ratios of successive terms approach the golden ratio, and in fact the terms themselves are rounding of integer powers of the golden ratio. The sequence also has a variety of relationships with the Fibonacci numbers, like the fact that adding any two Fibonacci numbers two terms apart in the Fibonacci sequence results in the Lucas number in between [1].

2. RELATED WORK

Many mathematics research scholars have done extensive research and have changed the way we use Lucas series for checking prime or not. The Lucas series is not only used for checking prime numbers but also mathematicians used for finding the next world's largest prime number. There are various softwares running on a server to find the next world's largest prime number which are running all over the world by millions of systems.

3. METHODOLOGY

The Lucas Mathematical Model (LMM) approach is used for checking a prime number, The Lucas numbers (ln) will be generated up to the required number of terms and stored in the secondary storage device in the form of text file, to store Lucas number in text file a python program is written and later it will be used thorough another python program meaning the program uses the text file as an input and check whether the given number entered thorough keyboard is prime or not.

To generate Lucas numbers(ln) the system ran the program for one days (24 hours) and it has generated the Lucas series (ls) which has a total input of 10^6-1 is stored in a file. It is observed that for input of 10^6-1, it has taken 1GB of space in the secondary storage device.

Further to check the efficiency of Lucas Mathematical Model (LMM) approach with the normal methodology of checking the prime numbers and examining the length of the input.

SJIF Impact Factor 2022: 8.197| ISI I.F. Value:1.241| Journal DOI: 10.36713/epra2016 ISSN: 2455-7838(Online)
EPRA International Journal of Research and Development (IJRD)
Volume: 7 | Issue: 4 | April 2022 - Peer Reviewed Journal

The various programming languages has the capability to generate Lucas number(ln) and store it in a file (permanently on secondary storage device). During the course of research, it was decided to use python programming language due to its large collection of library modules and the availability of online support.

ALGORITHM FOR STORING LUCAS NUMBERS ON SECONDARY STORAGE DEVICE

```
STEP 01: START
STEP 02: OPEN LUCAS FILE
STEP 03: READ LUCAS AS LUCAS1
STEP 04: LUCAS1=LIST (LUCAS)
STEP 05: A=1
STEP 06: B=3
STEP 07: WRITE A IN LUCAS FILE
STEP 08: WRITE B IN LUCAS FILE
STEP 09: INPUT N
STEP 10: FOR I IN RANGE (9*N):
STEP 11:         P=A+B
STEP 12:    WRITE P IN LUCAS FILE
STEP 13:    A=B
STEP 14:    B=P
STEP 15: CLOSE LUCAS FILE
STEP 16: STOP
```

PYTHON PROGRAM TO GENERATE THE LUCAS NUMBER AND STORING IN SECONDARY STORAGE DEVICE IN THE FORM OF TEXT FILE

```
lucas=open('lucas.txt','r+')
lucas1=lucas.readlines()
lucas1=list(lucas1)
a=1
b=3
n=int(input(‘Enter no of digits’)
lucas.write(str(a)+' ')
lucas.write(str(b)+' ')
for i in range(int('9'*5)):
    p=a+b
      lucas.write(str(p)+' ')
    a=b
    b=p
lucas.close()
```

ALGORITHM TO FIND A PRIME NUMBER IN A LUCAS FILE

```
STEP 01: START
STEP 02: OPEN LUCAS FILE IN R
MODE
STEP 03:READ THE NUMBERS
STEP 04:SPLIT THE NUMBERS
STEP 05: READ SEARCHING PRIME NO
STEP 06: CHECK THE NO 2 OR 3 OR 5
STEP 07: PRINT “IT IS PRIME NO”
STEP 08: ELSE CHECK NO MOD 2 OR
         NO MOD 3 OR NO MOD 5= 0
STEP 09:PRINT “IT IS NOT PRIME
         NO”
STEP 10: ELSE FIND LENGHT OF NO
AND STORE IN S VARIABLE
STEP 11:CHECK S<6
STEP 12: IF TRUE THEN CHECK IN
LUCAS FILE
STEP 13: SEARCH FOR THE POSITION
OF NUMBER IN LUCAS FILE
AND APPLYING MODULAS OPERATOR TO CHECK
FOR A PRIME IF IT IS TRUE PRINT PRIME
ELSE PRINT NOT A PRIME.
STEP 14: THE SAME SET OF OPERATIONS
(STEP 02 TO STEP 13) WILL BE CARRIED FOR
FINDING TEN DIGIT NO
STEP 15: STOP
```

PYTHON PROGRAM TO CHECK PRIME OR NOT

```
r=open('lucas.txt','r+')
lucas=r.read()
lucas=lucas.split()
x=int(input('Enter the number:'))
if(x==2 or x==3 or x==5):
     print(x,'is a prime number')
elif (x%2==0 or x%3==0 or x%5==0):
     print(x,'is  not a prime no')
else:
  s=len(str(x))
  if (s<6):
    if (((int(lucas[x-1])-1))%x==0):
           print(x,'is a prime number')
    else:
      print(x,'is not a prime number')

  elif(s<11):
      r.close()
      r=open('lucas1.txt','r+')
      lucas=r.read()
      lucas=lucas.split()
      s=(x-(int('9'*5)+1))

      if (((int(lucas[s-1])-1))%x==0):
        print(x,'is a prime no')
      else:
        print(x,'is not a prime number')
      r.close()

else:
    print('lucas is not generated')
```

4. COMPLEXITY OF ALGORITHM

In computer science, analysis of algorithms is a very crucial part. It is important to find the most efficient algorithm for solving a problem. It is possible to have many algorithms to solve a problem, but the challenge here is to choose the most efficient one.[2]

There are multiple ways to design an algorithm, or considering which one to implement in an application. When thinking through this, it's crucial to consider the algorithm's **time complexity** and **space complexity**.[3]

5. SPACE COMPLEXITY

The space complexity of an algorithm is the amount of space (or memory) taken by the algorithm to run as a function of its input length, n. Space complexity includes both auxiliary space and space used by the input.[3]

Auxiliary space is the temporary or extra space used by the algorithm while it is being executed. Space complexity of an algorithm is commonly expressed using **Big (O(n))** notation.[3]

The Space complexity is ignored in this research paper, since the space complexity of particular problem is not considered so important.

6. TIME COMPLEXITY

The time complexity of an algorithm is the amount of time taken by the algorithm to complete its process as a function of its input length, n. The time complexity of an algorithm is commonly expressed using asymptotic notations:[3]

Big O - 0(n)
Big Theta - Θ(n)
Big Omega - Ω(n)

It's valuable for a programmer to learn how to compare performances of different algorithms and choose the best time-space complexity to solve a particular problem in the most efficient way possible.[3]

Big O notation is used in Computer Science to portrait the performance or complexity of an algorithm.

Big O specifically defines the worst-case scenario of an algorithm, and can be used to describe the execution time required or the space used (e.g. in memory or on disk) by an algorithm. here O stands for order of growth.

Big Theta(Θ) is used to represent the average case scenario of an algorithm and can be used to describe the execution time required or the space used (e.g. in memory or on disk) by an algorithm.

Big Omega (Ω)is used to represent the best-case scenario of an algorithm and can be used to describe the execution time required or the space used (e.g., in memory or on disk) by an algorithm.

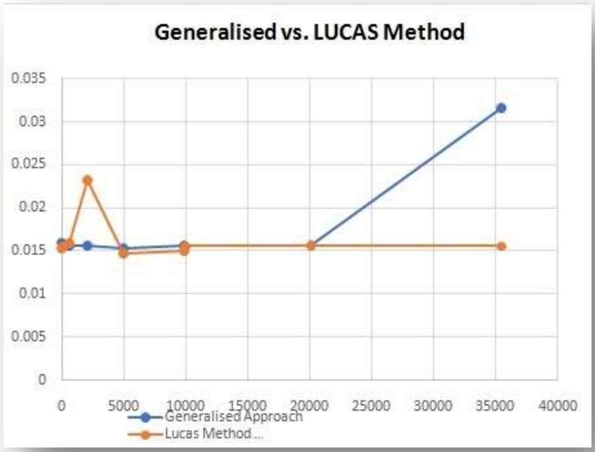
These three methods are the most common and very popular methods of design and analysis of an algorithm which are used for finding the efficiency of the program.

7. RUNTIME COMPLEXITY OF CHECKING A PRIME NUMBER

Input	Generalized approach	Lucas Mathematical Model(LMM)
7	0.016026497	0.015622139
97	0.015621662	0.015602868
643	0.015629292	0.015621185
2111	0.015626907	0.01562047
5003	0.015336752	0.016032696
9871	0.015622616	0.015021901
10007	0.015590906	0.015621424
20147	0.015659809	0.015634537
35507	0.031657457	0.015586376

Graphical Representation of Runtime complexity of both the methods

8.GENERALISED APPROACH - *rc*



In the normal approach the program checks for the given number prime or not. The time complexity of the algorithm for worst case is denoted as:

Big (O(n))

9. LUCAS METHOD (LMM) - *rc*

The time complexity of the LUCAS Method is calculated as

Big (O(14))

10. CONCLUSION

The Lucas mathematical methodology has the greater efficiency for checking prime when comparing with general approach. Further it is also observed that generating prime series and storing in a file is one time process and it is time consuming but once the file is prepared the performance of the code is much higher than the normal approach. In addition to this it is also observed that the execution of expression also depends on the hardware configuration.

11. ACKNOWLEDGEMENT

Apart from the efforts of me, the success of any work or project depends largely on the encouragement and guidelines of many others. I take this opportunity to express my gratitude to the people who have been instrumental in the successful completion of this research paper.

I express deep sense of gratitude to almighty God for giving me strength for the successful completion of the research paper.

I express my heartfelt gratitude to my parents for constant encouragement while carrying out this research paper.

I express my deep sense of gratitude to the luminary **The Principal Capt. (IN) Nirmal Raghu, Sainik School Amaravathinagar** who has been continuously motivating and extending their helping hand to us.

I express my sincere thanks to the academician The **Vice Principal Lt Col Nripendra Singh, Sainik School Amaravathinagar**, for constant encouragement and the guidance provided during this research.

My sincere thanks to **Mr. Praveen Kumar Murigeppa Jigajinni**, Master In-charge, A guide, Mentor and great motivator, who critically reviewed my paper and helped in solving each and every problem, occurred during implementation of this research paper.

12. REFERENCES

1. *https://en.wikipedia.org/wiki/Lucas_number*
2. *https://www.freecodecamp.org/news/time-complexity-of-algorithms/*
3. *https://www.educative.io/edpresso/time-complexity-vs-space-complexity*