



# WINTER SEMESTER

# 2015-2016

**TITLE OF THE PROJECT: AKS PRIMALITY  
TEST**

**STUDENT LIST (NAME & REGNO.):**

**Osho Agyeya(15BCE1326)**

**Shivam Prasad(15BCE1196)**

**Malladi Tejasvi(15BCE1208)**

**Shubankar B(15BCE1123)**

**Shivam Pratap Singh(15BCE1053)**

**Under the guidance of UMITTY  
SRINIVASA RAO, Work done at VIT  
University, Chennai**

**B.Tech Computer Science Engineering  
(2015-2016)**

**SCHOOL OF COMPUTING SCIENCE &  
ENGINEERING, VIT UNIVERSITY**

# INTRODUCTION:

The AKS primality test (also known as Agrawal–Kayal–Saxena primality test and cyclotomic AKS test) is a deterministic primality-proving algorithm created and published by Manindra Agrawal, Neeraj Kayal, and Nitin Saxena, computer scientists at the Indian Institute of Technology Kanpur, on August 6, 2002, in a paper titled "PRIMES is in P". The algorithm determines whether a number is prime or composite within polynomial time. The authors received the 2006 Gödel Prize and the 2006 Fulkerson Prize for this work. Significant aspects of this algorithm are:

- Invented by Manindra Agrawal, Neeraj Kayal & Nitin Saxena
- Unconditional: No constraints involved
- Deterministic: Solves the problem with exact decision at every step; no guesses involved; machine's current state determines what its next state will be; computes a mathematical function which has a unique value for any input in its domain
- Polynomial time complexity: Running time is upper bounded by a polynomial expression in the size of the input (no. of digits) for the algorithm

# OBJECTIVE:

The project focusses mainly on the following aspects:

- History of PRIMES
- Basic idea of AKS Primality Test
- Deduction of AKS algorithm from Fermat's theorem
- Main algorithm of AKS test
- Improvements made to AKS test
- Correctness of AKS test
- Time complexity analysis
- Implementation of AKS test
- Improvement in time complexity via our own ideas

# LITERATURE SURVEY:

Prior to this algorithm, various researches have been conducted on primality testing. We have included a few of them in our project.

Sieve of Eratosthenes: In mathematics, the **sieve of Eratosthenes**, one of a number of prime number sieves, is a simple, ancient algorithm for finding all prime numbers up to any given limit. It does so by iteratively marking as composite (i.e., not prime) the multiples of each prime, starting with the multiples of 2.

The multiples of a given prime are generated as a sequence of numbers starting from that prime, with constant difference between them that is equal to that prime. This is the sieve's key distinction from using trial division to sequentially test each candidate number for divisibility by each prime.

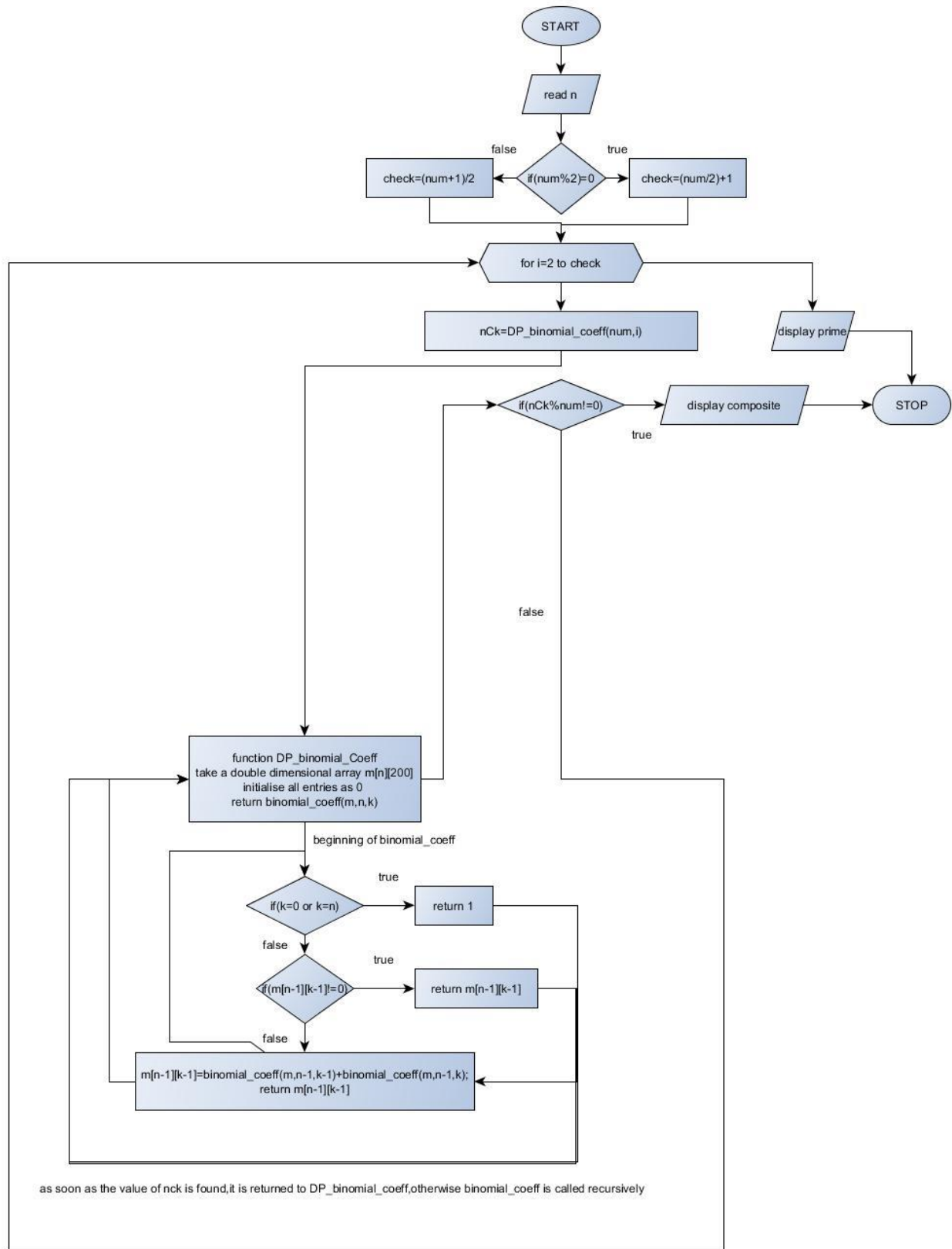
The sieve of Eratosthenes is one of the most efficient ways to find all of the smaller primes.

Fermat's theorem: **Fermat's little theorem** states that if  $p$  is a prime number, then for any integer  $a$ , the number  $a^p - a$  is an integer multiple of  $p$ . In the notation of modular arithmetic, this is expressed as

$$a^p \equiv a \pmod{p}.$$

For example, if  $a = 2$  and  $p = 7$ ,  $2^7 = 128$ , and  $128 - 2 = 7 \times 18$  is an integer multiple of 7.

# WORKFLOW/ARCHITECTURE DIAGRAM/SYSTEM DESIGN:



## METHODS AND TECHNOLOGIES:

The focal point of the algorithm is as follows:

The binomial coefficients of the no. input are generated. Since the binomial coefficients are symmetrically arranged, only half coefficients are generated. The binomial coefficients are generated by dynamic programming using top down approach. Now, each of these coefficients is checked for divisibility by the input number.  $nC_0=1$  is ignored for divisibility check. If all the coefficients are divisible by the input no, then the no. is prime. Otherwise, the no. is composite.

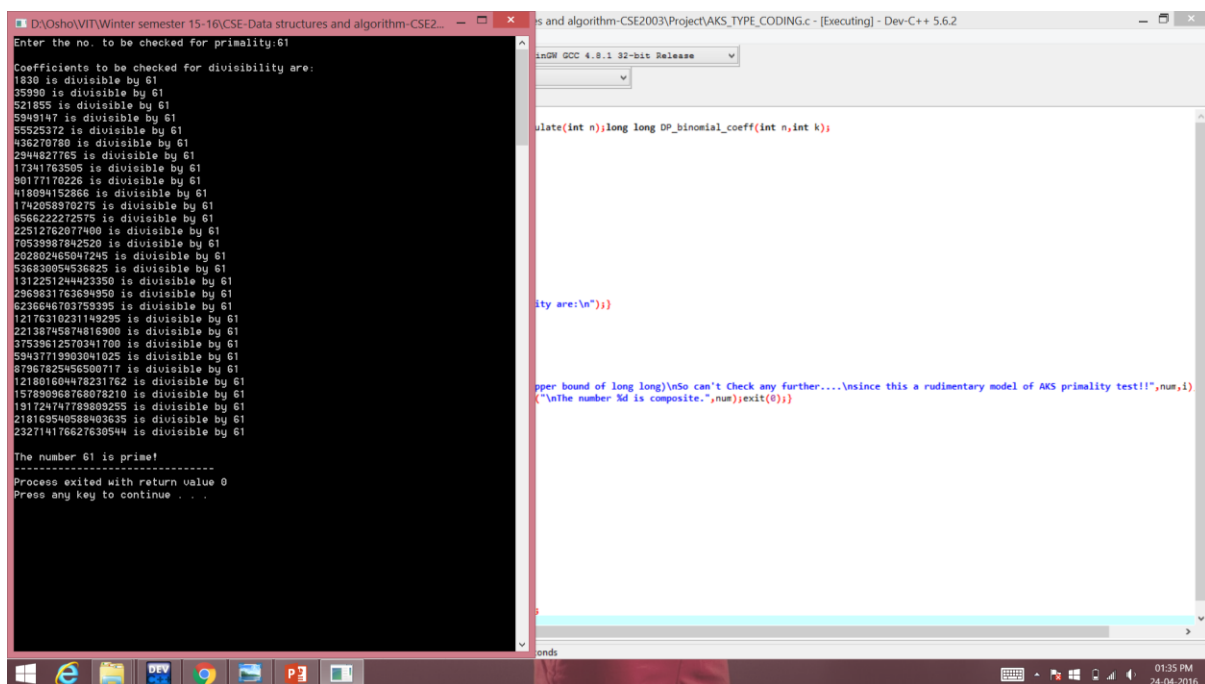
Let  $a$  be an integer &  $n$  be a natural number such that  $n \geq 2$ , HCF of  $a$  &  $n$  is 1. Then  $n$  is prime iff:

$(x+a)^n = x^n + a \pmod{n}$ , i.e. all the coefficients in  $((x+a)^n - (x^n + a))$  should be perfectly divisible by  $n$ .

In other words,  ${}^nC_i * a^{(n-i)}$  is perfectly divisible by  $n$  for  $0 < i < n$ .

## RESULTS AND OBSERVATIONS:

The algorithm developed by us tests the primality upto 61(due to the limitations of the datatype involved).The no. to be input cannot be 1,0 or any –ve integer.



The screenshot shows a Windows desktop with a Dev-C++ IDE running a C++ program. The program is titled "aks and algorithm-CSE2003\Project\AKS\_TYPE\_CODING.c - [Executing] - Dev-C++ 5.6.2". The console window on the left displays the following output:

```
Enter the no. to be checked for primality:61
Coefficients to be checked for divisibility are:
1830 is divisible by 61
35990 is divisible by 61
521855 is divisible by 61
5949147 is divisible by 61
55525372 is divisible by 61
436270789 is divisible by 61
2944827785 is divisible by 61
17341763905 is divisible by 61
90177170226 is divisible by 61
418094152866 is divisible by 61
1742058970275 is divisible by 61
9586222272575 is divisible by 61
22512762077409 is divisible by 61
70539987842520 is divisible by 61
202802465047245 is divisible by 61
936830954536825 is divisible by 61
1312251244423250 is divisible by 61
2969831763694950 is divisible by 61
6236646703759395 is divisible by 61
12176310231148295 is divisible by 61
22138745874818900 is divisible by 61
37529012570341700 is divisible by 61
59437719903041025 is divisible by 61
87967825456500717 is divisible by 61
121801604478231762 is divisible by 61
157890961768070210 is divisible by 61
19172477788009255 is divisible by 61
218169540588403635 is divisible by 61
232714176627630544 is divisible by 61

The number 61 is prime!
-----
Process exited with return value 0
Press any key to continue . . .
```

The code editor on the right shows the following C++ code:

```
#include <iostream>
using namespace std;

long long DP_binomial_coeff(int n,int k);

int main()
{
    int n;
    cout<<"Enter the no. to be checked for primality:"<<endl;
    cin>>n;
    if(n==1 || n==0 || n<0)
    {
        cout<<"Invalid input!"<<endl;
        return 0;
    }
    if(n==2 || n==3)
    {
        cout<<"The number is prime!"<<endl;
        return 0;
    }
    if(n%2==0 || n%3==0)
    {
        cout<<"The number is composite!"<<endl;
        return 0;
    }
    long long p=1;
    for(int i=2;i<=n;i++)
    {
        p*=i;
    }
    for(int i=2;i<=n;i++)
    {
        if(DP_binomial_coeff(n,i)%p!=0)
        {
            cout<<"The number is prime!"<<endl;
            return 0;
        }
    }
    cout<<"The number is composite!"<<endl;
    return 0;
}
```



## REFERENCES:

Introduction to Algorithms-Book by Charles E. Leiserson, Clifford Stein, Ronald Rivest, and Thomas H. Cormen

[www.wikipedia.com](http://www.wikipedia.com)

[www.youtube.com](http://www.youtube.com)

PRIMES is in P

Manindra Agrawal     Neeraj Kayal

Nitin Saxena

Department of Computer Science & Engineering

Indian Institute of Technology Kanpur

Kanpur-208016, INDIA

Email: {manindra,kayal,nitinsa}@iitk.ac.in

<http://mathworld.wolfram.com/AKSPrimalityTest.html>

<https://terrytao.wordpress.com/2009/08/11/the-aks-primality-test/>

Sanjoy Dasgupta, C.Papadimitriou and U.Vazirani , Algorithms , Tata McGraw-Hill, 2006.

A. V. Aho, J.E. Hopcroft and J. D. Ullman, Data Structures and Algorithms ,Pearson India, 1st Edition, 2002

Mark Allen Weiss, Data Structures and Algorithm Analysis in C, 2<sup>nd</sup> edition, Pearson Education, 2007

---