Primality Test | Set 4 (Solovay-Strassen)

We have already been introduced to primality testing in the previous articles in this series.

- Primality Test | Set 1 (Introduction and School Method)
- Primality Test | Set 2 (Fermat Method)
- Primality Test | Set 3 (Miller–Rabin)

The Solovay–Strassen primality test is a probabilistic test to determine if a number is composite or probably prime.
Before diving into the code we will need to understand some key terms and concepts to be able to code this algorithm.

Background:

Legendre Symbol: This symbol is defined for a pair of integers a and p such that p is prime. It is denoted by (a/p) and calculated as:

```
      = 0    if a%p = 0
(a/p) = 1    if there exists an integer k such that k^2 = a(modp)
      = -1   otherwise.
```

Euler proved that:

$(a/p) = a^{((p-1)/2)}$%p **Condition (i)**

Jacobian Symbol: This symbol is a generalization of Legendre Symbol, where p is replaced by n where n is

```
n = p1k1 * .. * pnkn
```

, then Jacobian symbol is defined as:

$(a/n) = ((a/p1)^{k1}) * ((a/p2)^{k2}) * ..... * ((a/pn)^{kn})$

If n is taken as a prime number then the Jacobian is equal to the Legendre symbol. These symbols have certain **properties** –
1) (a/n) = 0 if gcd(a,n) != 1, Hence (0/n) = 0. This is because if gcd(a,n) != 1, then there must be some prime pi such that pi divides both a and n. In that case (a/pi) = 0 [by definition of Legendre Symbol].
2) (ab/n) = (a/n) * (b/n). It can be easily derived from the fact (ab/p) = (a/p)(b/p) (here (a/p) is the Legendry Symbol).
3) If a is even, than (a/n) = (2/n)*((a/2)/n). It can be shown that:

```
      = 1 if n = 1 ( mod 8 ) or n = 7 ( mod 8 )
(2/n) = -1 if n = 3 ( mod 8 ) or n = 5 ( mod 8 )
      = 0 otherwise
```

4)

$(a/n) = (n/a)* (-1)^{((a-1)(n-1)/4))}$ if a and n are both odd.

The Algorithm:

We select a number n to test for its primality and a random number a which lies in the range of [2, n-1] and compute its Jacobian (a/n), if n is a prime number then the Jacobian will be equal to the Legendre and it will satisfy condition (i) given by Euler. If it does not satisfy the given condition then n is composite and the program will stop. Just like every other Probabilistic Primality Test its accuracy is also directly proportional to the number of iterations. So we run the test for several iterations to get more accurate results.

Note: We are not interested in calculating the Jacobian of even numbers as we already know that they are not prime except 2.

Pseudocode:

Algorithm for Jacobi:
Step 1 //base cases omitted
Step 2 if a>n then
Step 3 return $((amodn)/n)$
Step 4 else
Step 5 return $(-1)^{((a-1)/2)((n-1)/2)}(a/n)$
Step 6 endif

Algorithm for Solovay-Strassen:
Step 1 Pick a random element a < n
Step 2 if gcd(a, n) > 1 then
Step 3 return COMPOSITE
Step 4 end if
Step 5 Compute $a^{(n-1)/2}$ using repeated squaring and (a/n) using Jacobian algorithm.
Step 6 if (a/n) not equal to $a^{(n-1)/2}$ then
Step 7 return composite
Step 8 else

```
Step 9        return prime
Step 10   endif
```

Recommended: Please solve it on "*PRACTICE*" first, before moving on to the solution.

Implementation:

```
// C++ program to implement Solovay-Strassen
// Primality Test
#include <bits/stdc++.h>
using namespace std;

// modulo function to perform binary exponentiation
long long modulo(long long base, long long exponent,
                                    long long mod)
{
    long long x = 1;
    long long y = base;
    while (exponent > 0)
    {
        if (exponent % 2 == 1)
            x = (x * y) % mod;

        y = (y * y) % mod;
        exponent = exponent / 2;
    }

    return x % mod;
}

// To calculate Jacobian symbol of a given number
int calculateJacobian(long long a, long long n)
{
    if (!a)
        return 0;// (0/n) = 0

    int ans = 1;
    if (a < 0)
    {
        a = -a; // (a/n) = (-a/n)*(-1/n)
        if (n % 4 == 3)
            ans = -ans; // (-1/n) = -1 if n = 3 (mod 4)
    }

    if (a == 1)
        return ans;// (1/n) = 1

    while (a)
    {
        if (a < 0)
        {
            a = -a;// (a/n) = (-a/n)*(-1/n)
            if (n % 4 == 3)
                ans = -ans;// (-1/n) = -1 if n = 3 (mod 4)
        }

        while (a % 2 == 0)
        {
            a = a / 2;
            if (n % 8 == 3 || n % 8 == 5)
                ans = -ans;

        }

        swap(a, n);

        if (a % 4 == 3 && n % 4 == 3)
            ans = -ans;
```

```
        a = a % n;

        if (a > n / 2)
            a = a - n;

    }

    if (n == 1)
        return ans;

    return 0;
}

// To perform the Solovay-Strassen Primality Test
bool solovoyStrassen(long long p, int iterations)
{
    if (p < 2)
        return false;
    if (p != 2 && p % 2 == 0)
        return false;

    for (int i = 0; i < iterations; i++)
    {
        // Generate a random number a
        long long a = rand() % (p - 1) + 1;
        long long jacobian = (p + calculateJacobian(a, p)) % p;
        long long mod = modulo(a, (p - 1) / 2, p);

        if (!jacobian || mod != jacobian)
            return false;
    }
    return true;
}

// Driver Code
int main()
{
    int iterations = 50;
    long long num1 = 15;
    long long num2 = 13;

    if (solovoyStrassen(num1, iterations))
        printf("%d is prime\n",num1);
    else
        printf("%d is composite\n",num1);

    if (solovoyStrassen(num2, iterations))
        printf("%d is prime\n",num2);
    else
        printf("%d is composite\n",num2);

    return 0;
}
```

Run on IDE

Output:

```
15 is composite
13 is prime
```

Running Time: Using fast algorithms for modular exponentiation, the running time of this algorithm is O(k·log^3 n), where k is the number of different values of a we test.

Accuracy: It is possible for the algorithm to return an incorrect answer. If the input n is indee[illegible]e, then the output will always correctly be probably prime. However, if the input n is composite the[illegible] it is

possible for the output to be incorrectly probably prime. The number n is then called a Euler-Jacobi pseudoprime.

References:

- https://www.revolvy.com/topic/Solovay%E2%80%93Strassen%20primality%20test&item_type=topic
- https://www.wikiwand.com/en/Solovay%E2%80%93Strassen_primality_test
- http://www.cmi.ac.in/~ramprasad/lecturenotes/comp_numb_theory/lecture2021.pdf

This article is contributed by Palash Nigam . If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.