Criptografie-Tema3

guzurazvan

May 2024

24. Aplicați algoritmul lui Fermat pentru a determina primalitatea numărului 42307.

n=42307
Fie $a \in [2, n-1]$ un număr aleatoriu. Dacă $a^{n-1} = 1(modn)$ atunci n are o probabilitate mare să fie prim. În caz contrar, numarul n este compus.
Pentru $a = 2$ avem: $a^{n-1}(modn) = 2^{42306}(mod42307) = 1(mod42307)$. $\Rightarrow$ n = 42307 este cel mai probabil prim.

3. Arătați că dacă 2^n - 1 este prim, atunci n este prim.

Pt $n = 0 : 2^0 - 1 = 0$ (nu e număr prim).
Pt $n = 1 : 2^1 - 1 = 1$ (nu e număr prim).
Pt $n = 2 : 2^2 - 1 = 3$ (e prim, iar n = 2 e prim).
Pt $n = 3 : 2^3 - 1 = 7$ (e prim, iar n = 3 e prim).
Pentru n > 3:
P.p prin R.A. că nu nu e prim.
$\Rightarrow \exists a, b \in N^* - \{1\}$ a.i. $n = a \cdot b$
$2^n - 1 = 2^{a \cdot b} - 1 = 2^{ab} - 1 = (2^a - 1) \cdot ((2^a)^{b-1} + (2^a)^{b-2} + ... + (2^a)^1 + 1)$
$a > 1 \Rightarrow 2^a - 1 > 1$ (1)
$2^a < 2^n \Rightarrow 2^a - 1 < 2^n - 1$ (2)
Din (1) si (2) $2^n - 1$ are un divizor diferit de 1 și de el insuși $\Rightarrow 2^n - 1$ nu e prim. (Fals)
$\Rightarrow$ Dacă $2^n - 1$ este prim, atunci n este prim.

5. Implementați o funcție care să calculeze simbolul lui Jacobi

```
#include "Header.h"

int SimbolJacobi(int a, int n) {
    if (n <= 0 || n % 2 == 0) {
        cerr << "n trebuie sa fie un numar impar pozitiv." << endl;
        return 0;
```

```
    }
    a = modulo(a, n);
    int jac = 1;
    while (a != 0) {
        while (a % 2 == 0) {
            a /= 2;
            int n_mod_8 = n % 8;
            if (n_mod_8 == 3 || n_mod_8 == 5) {
                jac = -jac;
            }
        }
        swap(a, n);
        if (a % 4 == 3 && n % 4 == 3) {
            jac = -jac;
        }
        a = modulo(a, n);
    }
    return (n == 1) ? jac : 0;
}

int main() {
    int a, n;
    cout << "Introduce i valorile pentru a si n (n trebuie sa fie
        impar pozitiv): ";
    cin >> a >> n;

    int result = SimbolJacobi(a, n);
    cout << "Simbolul Jacobi (" << a << "/" << n << ") este " <<
        result << endl;

    return 0;
}
```

6. Implementați algoritmul Solovay – Strassen

```
#include "Header.h"

int SimbolJacobi(int a, int n) {
    if (n <= 0 || n % 2 == 0) {
        cerr << "n trebuie sa fie un numar impar pozitiv." << endl;
        return 0;
    }
    a = modulo(a, n);
    int jac = 1;
    while (a != 0) {
        while (a % 2 == 0) {
            a /= 2;
            int n_mod_8 = n % 8;
            if (n_mod_8 == 3 || n_mod_8 == 5) {
                jac = -jac;
            }
        }
        swap(a, n);
        if (a % 4 == 3 && n % 4 == 3) {
            jac = -jac;
        }
        a = modulo(a, n);
```

```
    }
    return (n == 1) ? jac : 0;
}

bool SolovayStrassen(int n, int k) {
    if (n < 2) return false;
    if (n != 2 && n % 2 == 0) return false;

    srand(time(0));
    for (int i = 0; i < k; i++) {
        int a = rand() % (n - 1) + 1;
        int x = SimbolJacobi(a, n);
        if (x == 0 || a_la_b_mod_c(a, (n - 1) / 2, n) != modulo(x,
            n)) {
            return false;
        }
    }
    return true;
}

int main() {
    int n, k;
    cout << "Introduceti numarul pentru testare: ";
    cin >> n;
    cout << "Introduceti numarul de incercari: ";
    cin >> k;

    if (SolovayStrassen(n, k)) {
        cout << n << " este probabil prim." << endl;
    }
    else {
        cout << n << " este compus." << endl;
    }
    return 0;
}
```