Київський національний університет імені Тараса Шевченка
Факультет комп’ютерних наук і кібернетики

Лабораторна робота №1
з курсу «Основи криптології»
на тему

«Обчислення функції найбільшого дільника та піднесення до спільного дільника»

Виконали:
Студенти групи ІПС-31
Андрєєв Ілля Євгенович
Безкровна Дар'я Олегівна
Вербицький Артем Віталійович
Дубина Андрій Володимирович
Павлюченко Василь Іванович

Київ
2025

Зміст

Бінарний НСД-алгоритм

Вхід: натуральні числа x, y, де $x \geq y$.

Вихід: *НСД(x, y)*.

Метод:

1. *d := 1;*
2. *while x і y are even do (x := x/2; y := y/2; d := 2d);*
3. *while x 6 = 0 do*

3.1. *while x is even do x := x/2;*

3.2. *while y is even do y := y/2;*

3.3. *t := |x − y|/2;*

3.4. *if x ≥ y then x := t else y := t;*

4. *return (d · y).*

Реалізація на C++:

```
int binary_gcd(int x, int y) {
    cout << " x = " << x << ", y = " << y << endl;

    // 1. d := 1;
    int d = 1;
    cout << "1st step: d = " << d << endl;

    // 2. while x та y парні
    while (x % 2 == 0 && y % 2 == 0) {
        x = x / 2;
        y = y / 2;
        d = d * 2;
        cout << "2nd step: x = " << x << ", y = " << y << ", d = " << d
<< endl;
    }

    // 3. основний цикл
    while (x != 0) {
        // 3.1: ділимо x, поки він парний
        while (x % 2 == 0) {
            x = x / 2;
            cout << "3.1: x/ 2 => x = " << x << endl;
        }

        // 3.2: ділимо y, поки він парний
        while (y % 2 == 0) {
```

```
            y = y / 2;
            cout << "3.2: y/ 2 => y = " << y << endl;
        }

        // 3.3: t := |x - y| / 2
        int t;
        if (x > y) {
            t = (x - y) / 2;
        }
        else {
            t = (y - x) / 2;
        }
        cout << "3.3: t = |" << x << " - " << y << "| / 2 = " << t <<
endl;

        // 3.4: замінити більше число на t
        if (x >= y) {
            x = t;
            cout << "3.4: x >= y => x = t => x = " << x << endl;
        }
        else {
            y = t;
            cout << "3.4: x < y => y = t => y = " << y << endl;
        }
    }

    // 4. результат
    int result = d * y;
    cout << "4th step: NDS = d * y = " << d << " * " << y << " = " <<
result << endl;

    return result;
}
```

Реалізація на Python:

```
def binary_gcd(x, y):
    # 1. d := 1;
    d = 1

    # 2. while x и y парні do (x := x / 2; y := y / 2; d := 2d);
    while x % 2 == 0 and y % 2 == 0:
        x = x // 2
        y = y // 2
        d = d * 2

    # 3. while x ≠ 0 do
    while x != 0:
        # 3.1 while x парне do x := x / 2;
        while x % 2 == 0:
            x = x // 2
```

```
        # 3.2 while y парне do y := y / 2;
        while y % 2 == 0:
            y = y // 2

        # 3.3 t := |x - y| / 2;
        if x > y:
            t = (x - y) // 2
        else:
            t = (y - x) // 2

        # 3.4 if x ≥ y then x := t else y := t;
        if x >= y:
            x = t
        else:
            y = t

    # 4. return (d · y);
    return d * y
```

Тестові приклади:

```
Enter the first number: 56
Enter the second number: 98
Initial values: x = 56, y = 98
Step 1: d = 1
Step 2: x = 28, y = 49, d = 2
Step 3.1: x is even => x = 14
Step 3.1: x is even => x = 7
Step 3.3: t = |7 - 49| / 2 = 21
Step 3.4: x < y => y = 21
Step 3.3: t = |7 - 21| / 2 = 7
Step 3.4: x < y => y = 7
Step 3.3: t = |7 - 7| / 2 = 0
Step 3.4: x >= y => x = 0
Step 4: GCD = d * y = 2 * 7 = 14
RESULT: GCD(56, 98) = 14
```

```
Enter the first number: 48
Enter the second number: 24
Initial values: x = 48, y = 24
Step 1: d = 1
Step 2: x = 24, y = 12, d = 2
Step 2: x = 12, y = 6, d = 4
Step 2: x = 6, y = 3, d = 8
Step 3.1: x is even => x = 3
Step 3.3: t = |3 - 3| / 2 = 0
Step 3.4: x >= y => x = 0
Step 4: GCD = d * y = 8 * 3 = 24
RESULT: GCD(48, 24) = 24
```

```
Enter the first number: 270
Enter the second number: 192
Initial values: x = 270, y = 192
Step 1: d = 1
Step 2: x = 135, y = 96, d = 2
Step 3.2: y is even => y = 48
Step 3.2: y is even => y = 24
Step 3.2: y is even => y = 12
Step 3.2: y is even => y = 6
Step 3.2: y is even => y = 3
Step 3.3: t = |135 - 3| / 2 = 66
Step 3.4: x >= y => x = 66
Step 3.1: x is even => x = 33
Step 3.3: t = |33 - 3| / 2 = 15
Step 3.4: x >= y => x = 15
Step 3.3: t = |15 - 3| / 2 = 6
Step 3.4: x >= y => x = 6
Step 3.1: x is even => x = 3
Step 3.3: t = |3 - 3| / 2 = 0
Step 3.4: x >= y => x = 0
Step 4: GCD = d * y = 2 * 3 = 6
RESULT: GCD(270, 192) = 6
```

Розширений бінарний НСД-алгоритм

знаходить числа a, b, v такі, що $v = НСД(x, y)$ і $v = ax + by$,

де $a, b \in Z, v, x, y \in N$.

Вхід: натуральні числа x, y.

Вихід: числа $a, b \in Z$ такі, що $ax + by = v$, де $v = НСД(x, y)$.

Метод:

1. d := 1;
2. while x and y are even do
1.1. (x := x/2; y := y/2; d := 2d);
2. u := x; v := y; A := 1; B := 0; C := 0; D := 1;
3. while u is even do
4.1. u := u/2;
4.2. if A ≡ B ≡ 0 (mod 2) then (A := A/2; B := B/2)
3.1. else (A := (A + y)/2; B := (B − x)/2);
4. while v is even do
4.1. 5.1. v := v/2;
4.2. 5.2. if C ≡ D ≡ 0 (mod 2) then (C := C/2; D := D/2)
4.3. else (C := (C + y)/2; D := (D − x)/2);
5. if u ≥ v then (u := u − v; A := A − C; B := B − D)
5.1. else (v := v − u; C := C − A; D := D − B);
6. if u = 0 then (a := C; b := D; return(a, b, d · v))
6.1. else go to 4.

Реалізація на C++:

```
bool isEven(int x) {
    return x % 2 == 0;
}

int* extendedGcd(int x, int y) {
    // 1
    int d = 1;

    // 2
    while (isEven(x) && isEven(y)) {
        x /= 2;
        y /= 2;
```

```
        d *= 2;
    }

    // 3
    int u = x;
    int v = y;
    int A = 1;
    int B = 0;
    int C = 0;
    int D = 1;

    int i = 0; // used as a step counter
    std::cout << "step " << i++ << "; u = " << u << "; v = " << v << "; A = " << A << "; B = " << B << "; C = " << C << "; D = " << D << std::endl;
    do {
        // 4
        while (isEven(u)) {
            u /= 2;
            if (isEven(A) && isEven(B)) {
                A /= 2;
                B /= 2;
            } else {
                A = (A + y) / 2;
                B = (B - x) / 2;
            }
            std::cout << "step " << i++ << "; u = " << u << "; v = " << v << "; A = " << A << "; B = " << B << "; C = " << C << "; D = " << D << std::endl;
        }


        // 5
        while (isEven(v)) {
            v /= 2;
            if (isEven(C) && isEven(D)) {
                C /= 2;
                D /= 2;
            } else {
                C = (C + y) / 2;
                D = (D - x) / 2;
            }
            std::cout << "step " << i++ << "; u = " << u << "; v = " << v << "; A = " << A << "; B = " << B << "; C = " << C << "; D = " << D << std::endl;
        }

        // 6
        if (u >= v) {
            u -= v;
            A -= C;
            B -= D;
        } else {
```

```
            v -= u;
            C -= A;
            D -= B;
        }
        std::cout << "step " << i++ << "; u = " << u << "; v = " << v <<
"; A = " << A << "; B = " << B << "; C = " << C << "; D = " << D <<
std::endl;

        // 7
        if (u == 0) {
            int a = C;
            int b = D;

            static int result[] = {a, b, d * v};
            return result;
        }
    } while (u != 0);
    return 0;
}
```

Реалізація на Python:

```
def is_even(x):
    return x % 2 == 0

def extended_gcd(x, y):
    # 1
    d = 1

    # 2
    while is_even(x) and is_even(y):
        x //= 2
        y //= 2
        d *= 2

    # 3
    u, v = x, y
    A, B, C, D = 1, 0, 0, 1

    i = 0  # step counter
    print(f"step {i}; u = {u}; v = {v}; A = {A}; B = {B}; C = {C}; D =
{D}")

    while u != 0:
        # 4
        while is_even(u):
            u //= 2
            if is_even(A) and is_even(B):
                A //= 2
                B //= 2
            else:
```

```
                A = (A + y) // 2
                B = (B - x) // 2
            print(f"step {i}; u = {u}; v = {v}; A = {A}; B = {B}; C =
{C}; D = {D}")
            i += 1

        # 5
        while is_even(v):
            v //= 2
            if is_even(C) and is_even(D):
                C //= 2
                D //= 2
            else:
                C = (C + y) // 2
                D = (D - x) // 2
            print(f"step {i}; u = {u}; v = {v}; A = {A}; B = {B}; C =
{C}; D = {D}")
            i += 1

        # 6
        if u >= v:
            u -= v
            A -= C
            B -= D
        else:
            v -= u
            C -= A
            D -= B
        print(f"step {i}; u = {u}; v = {v}; A = {A}; B = {B}; C = {C}; D
= {D}")
        i += 1

    # 7
    return C, D, d * v
```

Тестові приклади:

```
Enter x = 693
Enter y = 609
step 0; u = 693; v = 609; A = 1; B = 0; C = 0; D = 1
step 1; u = 84; v = 609; A = 1; B = -1; C = 0; D = 1
step 2; u = 42; v = 609; A = 305; B = -347; C = 0; D = 1
step 3; u = 21; v = 609; A = 457; B = -520; C = 0; D = 1
step 4; u = 21; v = 588; A = 457; B = -520; C = -457; D = 521
step 5; u = 21; v = 294; A = 457; B = -520; C = 76; D = -86
step 6; u = 21; v = 147; A = 457; B = -520; C = 38; D = -43
step 7; u = 21; v = 126; A = 457; B = -520; C = -419; D = 477
step 8; u = 21; v = 63; A = 457; B = -520; C = 95; D = -108
step 9; u = 21; v = 42; A = 457; B = -520; C = -362; D = 412
step 10; u = 21; v = 21; A = 457; B = -520; C = -181; D = 206
step 11; u = 0; v = 21; A = 638; B = -726; C = -181; D = 206
v = 21 = GCD(693, 609), a = -181, b = 206
```

```
Enter x = 7
Enter y = 500
step 0; u = 7; v = 500; A = 1; B = 0; C = 0; D = 1
step 1; u = 7; v = 250; A = 1; B = 0; C = 250; D = -3
step 2; u = 7; v = 125; A = 1; B = 0; C = 375; D = -5
step 3; u = 7; v = 118; A = 1; B = 0; C = 374; D = -5
step 4; u = 7; v = 59; A = 1; B = 0; C = 437; D = -6
step 5; u = 7; v = 52; A = 1; B = 0; C = 436; D = -6
step 6; u = 7; v = 26; A = 1; B = 0; C = 218; D = -3
step 7; u = 7; v = 13; A = 1; B = 0; C = 359; D = -5
step 8; u = 7; v = 6; A = 1; B = 0; C = 358; D = -5
step 9; u = 7; v = 3; A = 1; B = 0; C = 429; D = -6
step 10; u = 4; v = 3; A = -428; B = 6; C = 429; D = -6
step 11; u = 2; v = 3; A = -214; B = 3; C = 429; D = -6
step 12; u = 1; v = 3; A = 143; B = -2; C = 429; D = -6
step 13; u = 1; v = 2; A = 143; B = -2; C = 286; D = -4
step 14; u = 1; v = 1; A = 143; B = -2; C = 143; D = -2
step 15; u = 0; v = 1; A = 0; B = 0; C = 143; D = -2
v = 1 = GCD(7, 500), a = 143, b = -2
```

Алгоритм обчислення $a^d \pmod m$

Вхід: числа a, d, m.

Вихід: число $y = a^d \pmod m$.

Метод:

1. *Записати число d у двійковій системі числення* $d = d_0 d_1 \cdots d_{r-1} d_r$;
2. $y := 1; s := a;$
3. *for* $i = 0, 1, \ldots, r$ *do*
3.1. *if* $d_i = 1$ *then* $y := y \cdot s \pmod m;$
3.2. $s := s \cdot s \pmod m;$
4. *return(y).*

Реалізація на C++:

```
// Функція для піднесення a^d (mod m) згідно з другим алгоритмом
int modularExponentiation(int a, int d, int m) {
    int y = 1; // Ініціалізуємо y = 1
    int s = a; // s = a

    // Проходимо по кожному біту числа d
    for (int i = 0; d > 0; i++) {
        if (d & 1) { // Якщо поточний біт d_i дорівнює 1
            y = (1LL * y * s) % m;
        }
        s = (1LL * s * s) % m; // Підносимо s до квадрату за модулем
        d >>= 1; // Зсуваємо d вправо (видаляємо оброблений біт)
    }

    return y; // Повертаємо результат
}

// Функція для вимірювання часу виконання (бенчмарк)
void benchmark(int a, int d, int m, int runs) {
    auto start = high_resolution_clock::now();

    int result;
    for (int i = 0; i < runs; i++) {
        result = modularExponentiation(a, d, m);
    }

    auto end = high_resolution_clock::now();
    auto duration = duration_cast<microseconds>(end - start).count(); //
мікросекунди
```

```
    cout << "Час виконання: " << duration / double(runs) << " мкс" << endl;
    cout << "Результат: " << result << endl;
}
```

Реалізація на Python:

```
# Функція для обчислення (a^d) % m
def modular_exponentiation(a, d, m):
    y = 1
    s = a

    while d > 0:
        if d & 1:  # Перевіряємо, чи останній біт d дорівнює 1
            y = (y * s) % m
        s = (s * s) % m  # Квадрат числа s за модулем
        d >>= 1  # Зсув праворуч (відкидаємо останній біт)

    return y


# Функція для вимірювання часу виконання (бенчмарк)
def benchmark(func, *args, runs=1000):
    start = time.perf_counter()
    for _ in range(runs):
        result = func(*args)
    end = time.perf_counter()
    elapsed_time = (end - start) * 1_000_000  # мікросекунди
    return f"Час виконання: {elapsed_time:.2f} мкс", result
```

Тестові приклади:

```
Введіть значення a, d, m та кількість ітерацій: 52 23 7 10000
Час виконання: 7104.80 мкс, Результат: 5
Натисніть будь-яку клавішу для продовження...
```

```
=== МЕНЮ ===
1. Обчислити (a^d) % m
2. Виконати бенчмарк
3. Вихід
Ваш вибір: 1
Введіть значення a, d, m: 52 23 7
Результат: 5
```

Другий алгоритм обчислення $a^d \pmod m$

Вхід: числа a, d, m.

Вихід: число $y = a^d \pmod m$.

Метод:

1. *Записати число d у двійковій системі числення* $d = d_0 d_1 \cdots d_{r-1} d_r$;
2. $y := 1$;
3. *for* $i = r, r - 1, \ldots, 0$ *do*
3.1. $y := y \cdot y \pmod m$;
3.2. *if* $d_i = 1$ *then* $y := y \cdot a \pmod m$;
4. *return(y).*

Реалізація на C++:

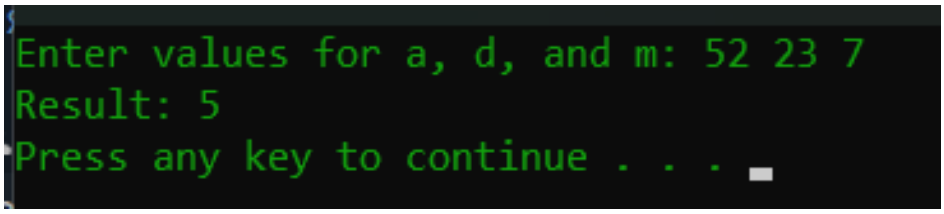
```
// Function to compute (a^d) % m
int modularExponentiation(int a, int d, int m) {
    int y = 1; // Initialize y = 1
    a = a % m; // Update a to a % m to reduce its value

    while (d > 0) {
        // If the current bit of d (d_0) is 1, multiply y by a and take
modulo
        if (d & 1) { // Check if the least significant bit is 1
            y = (1LL * y * a) % m; // Use 1LL to safely handle large
numbers
        }

        // Update 'a' to (a * a) % m using 1LL for large values
        a = (1LL * a * a) % m;

        // Perform a bitwise right shift on d
        d = d >> 1;
    }

    return y; // Return the result
}

// Benchmark wrapper function
template<typename ReturnType>
ReturnType benchmark(const function<ReturnType()>& func, int count =
1000, const string& description = "") {
    using namespace std::chrono;
    ReturnType res = NULL;
    // Start timing
```

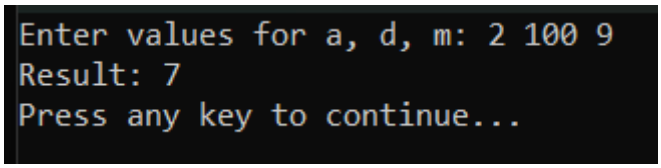
```
    auto start = high_resolution_clock::now();

    // Execute the function
    for(int i = 0; i < count; i++)
        res = func();

    // Stop timing
    auto stop = high_resolution_clock::now();

    // Calculate the duration in milliseconds
    auto duration = duration_cast<microseconds>(stop - start);

    // Output the result
    if (!description.empty()) {
        cout << "Benchmark for \"" << description << "\": ";
    }
    cout << "Execution time: " << duration.count() << " microseconds "
<< "or " << duration.count()/1000. << " ms" << endl;
    return res;
}
```

Реалізація на Python:

```
# Modular exponentiation implementation
def modular_exponentiation(a, d, m):
    y = 1
    a = a % m
    while d > 0:
        if d & 1:
            y = (y * a) % m
        a = (a * a) % m
        d = d >> 1
    return y

# Benchmarking function
def benchmark(func, count=1000):
    import time
    start = time.perf_counter()
    for _ in range(count):
        res = func()
    end = time.perf_counter()
    elapsed_time = end - start
    return f"Execution time: {elapsed_time * 1_000_000:.2f} microseconds
or {elapsed_time * 1_000:.2f} ms", res
```

Тестові приклади:





Частотний криптоаналіз

Зашифрований текст має вигляд:

UZQSOVUOHXMOPVGPOZPEVSGZWSZOPFPESXUDBMETSXAIZVU EPHZHMDZSHZOWSFPAPPDTSVPQUZWYMXUZUHSXEPYEPOPDZS ZUFPOMBZWPFUPZHMDJUDTMOHMQ

За допомогою таблиць відносної частоти появи літер та шляхом проб та помилок вдалося отримати такий текст:

IT WAS DISCLOSED YESTERDAY THAT SEVERAL INFORMAL BUT DIRECT CONTACTS HAVE BEEN MADE WITH POLITICAL REPRESENTATIVES OF THE VIET CONG IN MOSCOW.