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**

Бiнарний НСД-алгоритм ……………………………………………………….. 3

Розширений бiнарний НСД-алгоритм ………………………………………… 7

Алгоритм обчислення ad (mod m) …………………………………………….. 12

Другий алгоритм обчислення ad (mod m) ……………………………………. 14

Частотний криптоаналіз ………………………………………………………. 17

**Бiнарний НСД-алгоритм**

**Вхiд:** натуральнi числа *x, y*, де *x ≥ y*.

**Вихiд:** *НСД(x, y)*.

**Метод:**

1. *d := 1;*
2. *while x i y are even do (x := x/2; y := y/2; d := 2d);*
3. *while x 6 = 0 do*
   1. *while x is even do x := x/2;*
   2. *while y is even do y := y/2;*
   3. *t := |x − y|/2;*
   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:**

A black screen with white text

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

**Розширений бiнарний НСД-алгоритм**

знаходить числа *a*, *b*, *v* такi, що *v = НСД(x, y)* i *v = ax + by*,

де *a, b ∈ Z, v, x, y ∈ N*.

**Вхiд:** натуральнi числа *x, y*.

**Вихiд:** числа *a, b ∈ Z* такi, що *ax + by = v*, де *v = НСД(x, y)*.

**Метод:**

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

A black background with white text

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

**Алгоритм обчислення ad (mod m)**

**Вхiд:** числа *a, d, m*.

**Вихiд:** число *y = ad (mod m)*.

**Метод:**

1. *Записати число d y двiйковiй системi числення d = d0d1 · · · dr−1dr ;*
2. *y := 1; s := a;*
3. *for i = 0, 1, . . . , r do*
   1. *if di = 1 then y := y · s (mod m);*
   2. *s := s · s (mod 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:**

A black screen with white text

AI-generated content may be incorrect.

A black screen with white text

AI-generated content may be incorrect.

**Другий алгоритм обчислення ad (mod m)**

**Вхiд:** числа *a, d, m*.

**Вихiд:** число *y = ad (mod m)*.

**Метод:**

1. *Записати число d у двiйковiй системi числення d = d0d1 · · · dr−1dr ;*
2. *y := 1;*
3. *for i = r, r − 1, . . . , 0 do*
   1. *y := y · y (mod m);*
   2. *if di = 1 then y := y · a (mod 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:**

A screenshot of a computer

AI-generated content may be incorrect.

**A black background with white text

AI-generated content may be incorrect.**

**Частотний криптоаналіз**

Зашифрований текст має вигляд:

**UZQSOVUOHXMOPVGPOZPEVSGZWSZOPFPESXUDBMETSXAIZVUEPHZHMDZSHZOWSFPAPPDTSVPQUZWYMXUZUHSXEPYEPOPDZSZUFPOMBZWPFUPZHMDJUDTMOHMQ**

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.**