Київський національний університет імені Тараса Шевченка

Факультет комп`ютерних наук та кібернетики

Кафедра інтелектуальних програмних систем

Математичні основи захисту інформації

Лабораторна робота №3

Варіант №2

Виконали студенти 4-го курсу

Групи ІПС-42

Пащенко Дмитро Вікторович

Бондарець Дарина Володимирівна

Мединський Микола Олександрович

Київ - 2022

Завдання

Виконати роботу на тему: Обчислення a^d $(mod\ p)$ зліва направо і справа наліво.

- Розібратись, як працює алгоритм.
- Оцінити арифметичну складність цього алгоритму.
- Написати програму для обчислення a^d $(mod\ p)$ зліва направо і справа наліво.
- Описати 3-4 приклади, які демонструють роботу програми.
- Виконану роботу описати у звіті.

Ролі виконавців

Пащенко Дмитро Вікторович:

- Написав програму для обчислення a^d $(mod\ p)$ зліва направо і справа наліво.

Бондарець Дарина Володимирівна:

- Оцінила арифметичну складність даних алгоритмів.
- Виконану роботу описала у звіті.

Мединський Микола Олександрович:

- Описав 4 приклади, які демонструють роботу програми.

Теорія

Однією з найважливіших операцій на асиметричної криптографії є зведення числа на ступінь. Так як операція виконується в кінцевому полі, то фактично завдання зводиться до знаходження a^d *(mod p)*. Очевидно, що найпростішим способом розв'язання є виконання *d-1* множення, однак такий спосіб неприйнятний, коли йдеться про числа великої розрядності. Ефективні методи виконання зведення числа у ступінь за модулем будуть розглянуті у цій роботі.

Існує два способи зменшення часу виконання операції зведення у ступінь у кінцевому полі. По-перше, це зменшення часу виконання множення двох елементів групи. Другим способом є зменшення кількості операцій множення. В ідеалі обидва підходи мають бути використані одночасно.

Як рішення, що використовує зменшення кількості операцій множення при зведенні в ступінь, можна навести наступний алгоритм:

Алгоритм обчислення a^d *(mod p)* справа наліво

Вхід: Числа *a, d, p*.

Вихід: Число $y = a^d$ *(mod p)*

Метод:

1. Записати число *d* у двійковій системі числення $d = d_0d_1 \dots d_{r-1}d_r$.
2. *y := 1; s := a;*
3. for *i = 0, 1, ... , r* do
   3.1. if $d_i = 1$ then *y := y * s (mod p)*;
   3.2. *s := s * s (mod p);*
4. return(*y*).

Алгоритм обчислення a^d *(mod p)* зліва направо

Вхід: Числа *a, d, p*.

Вихід: Число $y = a^d$ *(mod p)*

Метод:

5. Записати число *d* у двійковій системі числення $d = d_0 d_1 \dots d_{r-1} d_r$.
6. *y := 1;*
7. for $i = r, r-1, \dots, 0$ do
   3.1. *y := y * y (mod p);*
   3.2. if $d_i = 1$ then *y := y * a (mod p)*;
8. return(*y*).

Оцінка арифметичної складності алгоритму

Часова складність даного алгоритму обчислення a^d *(mod p)* зліва направо і справа наліво складає $O(\log_2 d)$. Складність по пам’яті складатиме O(1).

Перелік основних модулів програми з коментарями

Клас, що містить функції для швидкого піднесення в степінь

public abstract class RepeatedSquaringModularExponentiation

Функція цього класу для швидкого піднесення в степінь методом справа-наліво

public static BigInteger powRightToLeft(BigInteger number, BigInteger power, BigInteger modulo, PrintStream out)

Функція цього класу для швидкого піднесення в степінь методом зліва-направо

public static BigInteger powLeftToRight(BigInteger number, BigInteger power, BigInteger modulo, PrintStream out)

Приклади

Приклад №1. (справа наліво)

```
EXAMPLE: 7 ^ 560 (mod 561) = 1

RIGHT TO LEFT

560_2 = 1000110000

Current bit: 0
7^2 (mod 561) = 49

Current bit: 0
49^2 (mod 561) = 157

Current bit: 0
157^2 (mod 561) = 526

Current bit: 0
526^2 (mod 561) = 103

Current bit: 1
1*103 (mod 561) = 103
103^2 (mod 561) = 511

Current bit: 1
103*511 (mod 561) = 460
511^2 (mod 561) = 256

Current bit: 0
256^2 (mod 561) = 460

Current bit: 0
460^2 (mod 561) = 103

Current bit: 0
103^2 (mod 561) = 511

Current bit: 1
460*511 (mod 561) = 1
511^2 (mod 561) = 256

RESULT: 1
```

Приклад №2. (зліва направо)

```
EXAMPLE: 7 ^ 560 (mod 561) = 1

LEFT TO RIGHT

560_2 = 1000110000

Current bit: 1
1^2 (mod 561) = 1
1*7 (mod 561) = 7

Current bit: 0
7^2 (mod 561) = 49

Current bit: 0
49^2 (mod 561) = 157

Current bit: 0
157^2 (mod 561) = 526

Current bit: 1
526^2 (mod 561) = 103
103*7 (mod 561) = 160

Current bit: 1
160^2 (mod 561) = 355
355*7 (mod 561) = 241

Current bit: 0
241^2 (mod 561) = 298

Current bit: 0
298^2 (mod 561) = 166

Current bit: 0
166^2 (mod 561) = 67

Current bit: 0
67^2 (mod 561) = 1

RESULT: 1
```

Приклад №3. (справа наліво)

```
EXAMPLE: 595 ^ 703 (mod 991) = 1


RIGHT TO LEFT

703_2 = 1010111111

Current bit: 1
1*595 (mod 991) = 595
595^2 (mod 991) = 238

Current bit: 1
595*238 (mod 991) = 888
238^2 (mod 991) = 157

Current bit: 1
888*157 (mod 991) = 676
157^2 (mod 991) = 865

Current bit: 1
676*865 (mod 991) = 50
865^2 (mod 991) = 20

Current bit: 1
50*20 (mod 991) = 9
20^2 (mod 991) = 400

Current bit: 1
9*400 (mod 991) = 627
400^2 (mod 991) = 449

Current bit: 0
449^2 (mod 991) = 428

Current bit: 1
627*428 (mod 991) = 786
428^2 (mod 991) = 840

Current bit: 0
840^2 (mod 991) = 8

Current bit: 1
786*8 (mod 991) = 342
8^2 (mod 991) = 64

RESULT: 342
```

Приклад №4. (зліва направо)

```
EXAMPLE: 595 ^ 703 (mod 991) = 1


LEFT TO RIGHT

703_2 = 1010111111

Current bit: 1
1^2 (mod 991) = 1
1*595 (mod 991) = 595

Current bit: 0
595^2 (mod 991) = 238

Current bit: 1
238^2 (mod 991) = 157
157*595 (mod 991) = 261

Current bit: 0
261^2 (mod 991) = 733

Current bit: 1
733^2 (mod 991) = 167
167*595 (mod 991) = 265

Current bit: 1
265^2 (mod 991) = 855
855*595 (mod 991) = 342

Current bit: 1
342^2 (mod 991) = 26
26*595 (mod 991) = 605

Current bit: 1
605^2 (mod 991) = 346
346*595 (mod 991) = 733

Current bit: 1
733^2 (mod 991) = 167
167*595 (mod 991) = 265

Current bit: 1
265^2 (mod 991) = 855
855*595 (mod 991) = 342

RESULT: 342
```

Перелік літературних джерел

- «Математичні основи захисту інформації.» С.Л. Кривий, ст. 108
- https://en.wikipedia.org/wiki/Modular_exponentiation