СПЕЦІАЛЬНІ РОЗДІЛИ

ОБЧИСЛЮВАЛЬНОЇ МАТЕМАТИКИ

КОМП’ЮТЕРНИЙ ПРАКТИКУМ No3

Багаторозрядна арифметика

1. Мета роботи

Одержання практичних навичок програмної реалізації обчислень у полі Галуа

характеристики 2 в поліноміальному базисі; ознайомлення з прийомами ефективної реалізації

критичних по часу ділянок програмного коду та методами оцінки їх ефективності.

2. Завдання до комп’ютерного практикуму

Реалізувати операції:

1) знаходження константи 0 – нейтрального елемента по операції «+»;

2) знаходження константи 1 – нейтрального елемента по операції «»;

3) додавання елементів;

4) множення елементів;

5) обчислення сліду елементу;

6) піднесення елемента поля до квадрату;

7) піднесення елемента поля до довільного степеня

8) знаходження оберненого елемента за множенням;

9) конвертування (переведення) елемента поля в m -бітний рядок (строкове зображення) і навпаки

Перевірити коректність операцій та знайти визначити середній час виконання

3. Код :

from mul\_matrix import matrix

class GaloisFieldPolynom:

    deg = 431

    matrix = matrix

    def \_\_init\_\_(self, vector: str) -> None:

        vector = vector.lstrip('0')

        vector = vector[-431:]

        for bit in vector:

            if bit not in '01':

                raise ValueError('Invalid polynom format')

        self.vector = '0' \* (self.deg - len(vector)) + vector

    def \_\_add\_\_(self, other: 'GaloisFieldPolynom') -> 'GaloisFieldPolynom':

        result = ''

        for x, y in zip(self.vector, other.vector):

            if x == y:

                result += '0'

            else:

                result += '1'

        return GaloisFieldPolynom(result)

    def \_\_mul\_\_(self, other):

        result = ''

        for i in range(self.deg):

            result += self.mul\_mtr(self << i, other << i)

        return GaloisFieldPolynom(result)

    def mul\_mtr(self, left, right):

        first = ''

        for i in range(self.deg):

            cell = 0

            for j in range(self.deg):

                a = int(left.vector[j])

                b = int(self.matrix[i][j])

                cell += a\*b

            first += str(cell % 2)

        result = 0

        for i in range(self.deg):

            result += int(first[i]) \* int(right.vector[i])

        return str(result % 2)

    def trace(self):

        result = GaloisFieldPolynom('0')

        prev = self

        for \_ in range(self.deg):

            val = prev\*prev

            result += val

            prev = val

        return result

    def inverse(self):

        return self\*\*((2\*\*self.deg) - 2)

    def \_\_pow\_\_(self, pow):

        result = GaloisFieldPolynom('1'\*431)

        for \_ in range(pow):

            result \*= self

        return result

    def \_\_lshift\_\_(self, value):

        vector = self.vector[value:] + self.vector[:value]

        return GaloisFieldPolynom(vector)

    def \_\_rshift\_\_(self, value):

        vector = self.vector[-value:] + self.vector[:-value]

        return GaloisFieldPolynom(vector)

    def \_\_eq\_\_(self, other):

        return self.vector == other.vector

    def \_\_ne\_\_(self, other):

        return self.vector != other.vector

    def \_\_repr\_\_(self) -> str:

        repr = self.vector.lstrip('0')

        if repr:

            return repr

        return '0'

4. Перевірка на коректність:

for i in range(10):

    a = GaloisFieldPolynom(''.join(choices('01', k=128)))

    b = GaloisFieldPolynom(''.join(choices('01', k=128)))

    c = GaloisFieldPolynom(''.join(choices('01', k=3)))

    if (a+b)\*c != (a\*c + b\*c):

        raise Exception

    if a\*\*(2\*\*a.deg - 1) != GaloisFieldPolynom('1'\*431):

        raise Exception

    result = GaloisFieldPolynom('1'\*431)

    for \_ in range(100):

        result \*= c

    if result != c\*\*100:

        raise Exception

    if c.inverse() \* c != GaloisFieldPolynom('1'\*431):

        raise Exception

Перевірка проходить без помилок

5. Визначення часу роботи:

start\_init = time()

for \_ in range(1000):

    num = GaloisFieldPolynom(''.join(choices('01', k=128)))

end\_init = time()

init = end\_init-start\_init

start\_inverse = time()

num = GaloisFieldPolynom(''.join(choices('01', k=128)))

num.inverse()

end\_inverse = time()

start\_trace = time()

num = GaloisFieldPolynom(''.join(choices('01', k=128)))

num.trace()

end\_trace = time()

start\_add = time()

for \_ in range(10):

    num1 = GaloisFieldPolynom(''.join(choices('01', k=128)))

    num2 = GaloisFieldPolynom(''.join(choices('01', k=128)))

    num1+num2

end\_add = time()

start\_mul = time()

for \_ in range(10):

    num1 = GaloisFieldPolynom(''.join(choices('01', k=128)))

    num2 = GaloisFieldPolynom(''.join(choices('01', k=128)))

    num1\*num2

end\_mul = time()

start\_pow = time()

for \_ in range(10):

    num1 = GaloisFieldPolynom(''.join(choices('01', k=128)))

    num1\*\*128

end\_pow = time()

init = end\_init - start\_init

inverse = (end\_inverse - start\_inverse- init/1000)

trace = (end\_trace - start\_trace - init/1000)

add = (end\_add - start\_add - init\*2/1000)/10

mul = (end\_mul - start\_mul - init\*2/1000)/10

pow = (end\_pow - start\_pow - init/1000)/10

Таблиця з отриманими результатами (час у секундах):

Пошук оберненого здійснюється за 2.34 години, а пошук сліду за 3.13 години

