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 :

class GaloisFieldPolynom:

    deg = 431

    gen = '101011'

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

        vector = vector.lstrip('0')

        for bit in vector:

            if bit not in '01':

                raise ValueError('Invalid polynom format')

        if len(vector) > self.deg:

            result = GaloisFieldPolynom(vector[-431:])

            for i in range(self.deg, len(vector)):

                if vector[::-1][i] == '1':

                    result += GaloisFieldPolynom(self.gen) << (i - self.deg)

            self.vector = result.vector

        else:

            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 \_\_lshift\_\_(self, value) -> 'GaloisFieldPolynom':

        return GaloisFieldPolynom(self.vector + '0' \* value)

    def \_\_eq\_\_(self, other):

        return self.vector == other.vector

    def \_\_ne\_\_(self, other):

        return self.vector != other.vector

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

        result = GaloisFieldPolynom('0')

        for i in range(len(other.vector)):

            if other.vector[i] == '1':

                result += (self << (len(other.vector) - i - 1))

        return result

    def \_\_pow\_\_(self, pow: int) -> 'GaloisFieldPolynom':

        result = GaloisFieldPolynom('1')

        pow = bin(pow)[2:]

        for bit in pow:

            result = result \* result

            if bit == '1':

                result = result \* self

        return result

    def trace(self):

        result = GaloisFieldPolynom('0')

        for i in range(self.deg):

            print(i)

            result = result + self \*\*(2\*\*i)

        return result

    def inverse(self):

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

    def \_\_len\_\_(self):

        return len(self.vector.lstrip('0'))

    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'):

        raise Exception

    result = GaloisFieldPolynom('1')

    for \_ in range(1000):

        result \*= c

    if result != c\*\*1000:

        raise Exception

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

        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 600 секунд, а пошук сліду за 685 секунд

