МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
(МОСКОВСКИЙ ПОЛИТЕХ)

Факультет информационных технологий
Кафедра «Инфокогнитивные технологии»

Лабораторная работа 4

По дисциплине «Защита информации»
Направление подготовки 09.03.03 «Прикладная информатика»
Профиль «Корпоративные информационные системы»

Выполнил:

студент группы 201-361

Погудин Александр

Москва 2023

Цель работы: написать программу, генерирующую и визуализирующую все решения уравнения вида y^2 ≡ x^3 + ax + b (mod p), где a, b ∈ Zp, где p – простое число (т.е. точки произвольной эллиптической кривой над конечным полем). Реализуйте операции (с наглядным представлением результата): 1) сложения двух точек кривой; 2) удвоения точки кривой.

Введение

Эллиптическая кривая - это математический объект, представляющий собой кривую в плоскости, заданную уравнением вида y^2 = x^3 + ax + b, где a и b - константы. Она получила свое название из-за своей формы, напоминающей эллипс. Одно из основных свойств эллиптических кривых - это то, что они обладают структурой группы. Это значит, что по определенному правилу можно складывать две точки на данной кривой и получать новую точку, также лежащую на этой кривой. Такие группы точек на эллиптических кривых широко используются в криптографии для создания криптографических протоколов и алгоритмов.

Программа

```python
import matplotlib.pyplot as plt

class Point:
    def __init__(self, x, y, a, b, p):
        self.x = x
        self.y = y
        self.a = a
        self.b = b
        self.p = p

        if self.x != None and self.y != None:
            if not ((self.y ** 2) % self.p == (self.x ** 3 + self.a * self.x + self.b) % self.p):
                raise ValueError("Точка не находится на кривой")


    def __add__(self, other):
        # Если мы складываем точку с самой собой, то удваиваем ее, используя метод double()
        if self.x == other.x and self.y == other.y:
            return self.double()

        # Если x-координаты двух точек равны, то результатом
        # сложения является точка (None, None)
        if self.x == other.x:
            return Point(None, None, self.a, self.b, self.p)

        # Вычисляем коэффициенты x и y
        s = (other.y - self.y) * pow(other.x - self.x, -1, self.p)
        x = (s ** 2 - self.x - other.x) % self.p
        y = (s * (self.x - x) - self.y) % self.p

        return Point(x, y, self.a, self.b, self.p)

    def double(self):
        # Вычисляем коэффициенты x и y
        s = (3 * self.x ** 2 + self.a) * pow(2 * self.y, -1, self.p)
        x = (s ** 2 - 2 * self.x) % self.p
        y = (s * (self.x - x) - self.y) % self.p

        return Point(x, y, self.a, self.b, self.p)

# Функция для нахождения точек, которые принадлежат кривой
get_points_on_curve = lambda a, b, p: [Point(x, y, a, b, p) for x in range(p) for y in range(p) if (y ** 2) % p == (x ** 3 + a * x + b) % p]
```

Данная программа реализует работу с точками на эллиптической кривой. В программе определен класс `Point` с методами `init()` и `add()`.

Метод `init()` принимает на вход пять аргументов: координаты точки `x` и `y`, коэффициенты эллиптической кривой `a` и `b`, а также модуль `p`. Внутри метода проверяется, что точка находится на кривой. Если условие не выполняется, то возникает исключение.

Метод `add()` реализует сложение двух точек. Для начала метод проверяет, что точки не совпадают и не являются противоположными (т.е. не имеют одинаковые координаты x, но разные координаты y). В зависимости от этого метод либо удваивает первую точку методом `double()`, либо находит координаты новой точки, результатом сложения которой является итоговая точка.

В методе `add()` используется математическая операция возведения в степень по модулю (`pow()`), что позволяет ускорить работу программы.

Метод `double()` также работает с точкой на эллиптической кривой. Он используется в случае, если метод `add()` получает на вход две одинаковые точки.

Метод вычисляет коэффициенты `x` и `y` новой точки, которая является результатом удвоения исходной.

Функция `get_points_on_curve` возвращает список точек на кривой, которые лежат в заданном диапазоне координат x и y. Для этого используется лямбда-функция, которая принимает на вход коэффициенты эллиптической кривой `a`, `b` и модуль `p`. Затем, в двойном цикле перебираются все возможные значения координат `x` и `y` в диапазоне от `0` до `p-1`. Для каждой точки вычисляется левая и правая часть уравнения кривой. Если они равны, то точка добавляется в список. В итоге функция возвращает список объектов типа `Point`, которые представляют собой точки на заданной кривой.

```
def display(points, fig, ax):
    # Область для отображения точек
    ax.set_xlim([-0.3, p])
    ax.set_ylim([-0.3, p])
    plt.grid(color = 'black', linewidth = 0.5)
    for point in points:
        ax.scatter(point.x, point.y, color = "k")

p = int(input("Введите p = ")) # 5
a = int(input("Введите a = ")) # 2
b = int(input("Введите b = ")) # 1

points = get_points_on_curve(a, b, p)

for point in points:
    print(f"({point.x}, {point.y})")

fig, ax = plt.subplots(1)
display(points, fig, ax)
```

Данный код отвечает за визуализацию списка точек на графике.

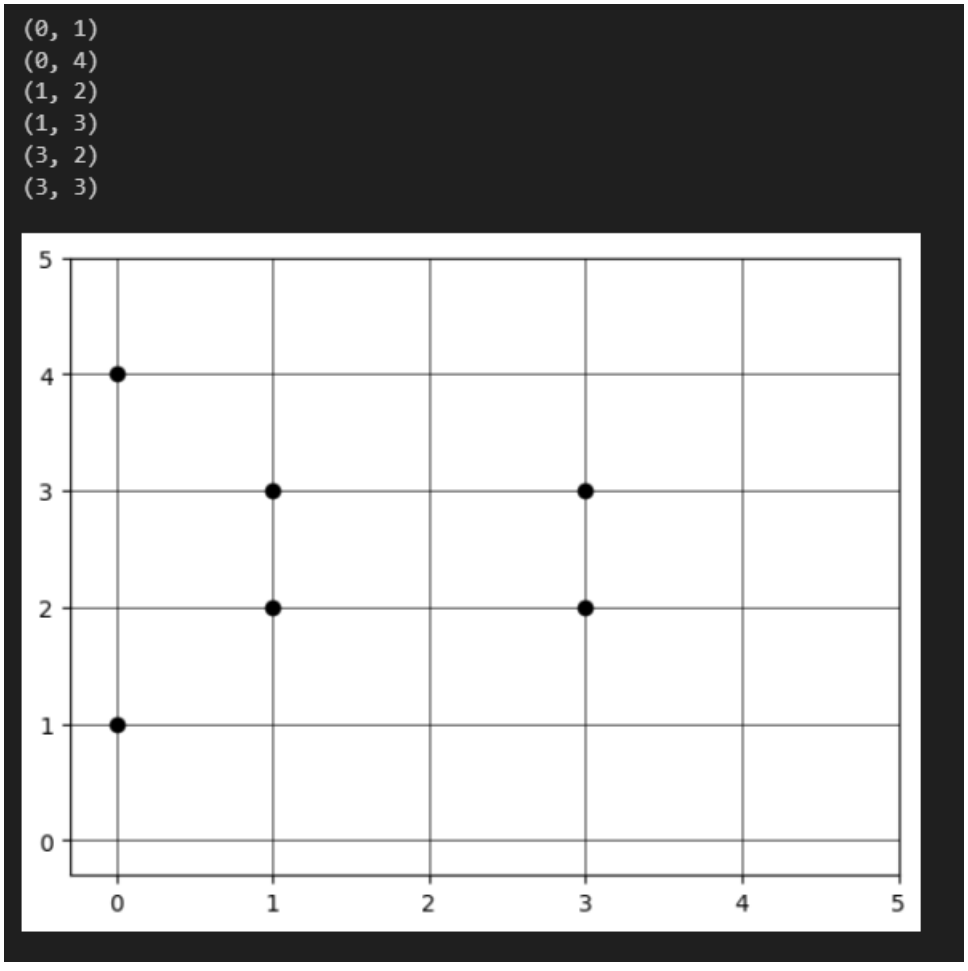
Сначала определяются границы области для отображения точек на графике при помощи методов `set_xlim()` и `set_ylim()` объекта `ax`. Затем задается сетка для графика с помощью функции `grid()`.

В цикле перебираются все точки из переданного списка `points`. Для каждой точки вызывается метод `scatter()` объекта `ax`, который отображает точку на графике.

После этого определяется модуль `p`, коэффициенты `a` и `b` заданные пользователем. Затем находятся все точки на кривой при помощи функции `get_points_on_curve()`, и их координаты выводятся на экран.

Наконец, создается объект `fig`, представляющий собой один изображения графика, и вызывается метод `subplots()`, чтобы нарисовать график. В конце вызывается функция `display()` для отображения точек на графике.

Результат работы программы, при p = 5, a = 2, b = 1.



```
fig, ax = plt.subplots(1)
display(points, fig, ax)

# Точки для сложения
print("Сложение точек")
point1 = Point(int(input("Введите x1 = ")), int(input("Введите y1 = ")), a, b, p)
point2 = Point(int(input("Введите x2 = ")), int(input("Введите y2 = ")), a, b, p)

# Сумма точек
sum_point = point1 + point2
print(f"Сумма двух точек ({point1.x}, {point1.y}) и ({point2.x}, {point2.y}): ({sum_point.x}, {sum_point.y})")
ax.scatter(sum_point.x, sum_point.y, color = "r")
if sum_point.x == None:
    print("Бесконечно удаленная точка")
```

В этой части кода определен блок инструкций, который позволяет пользователю находить сумму двух точек на заданной кривой.

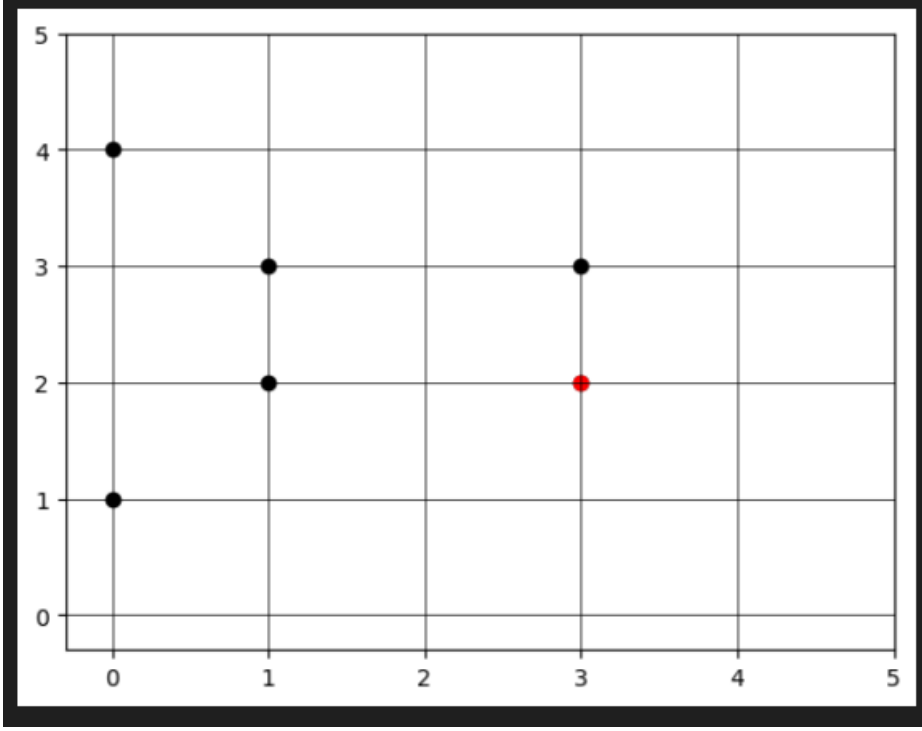
Создается объект `fig`, представляющий из себя одно изображение графика, и объект `ax`, который представляет собой область для рисования графика. Далее вызывается функция `display()`, которая отображает на графике все точки из списка `points`.

Далее пользователю предлагается ввести координаты двух точек, которые будут складываться, при помощи функций `input()`. Эти значения используются для создания объектов класса `Point` с именами `point1` и `point2`.

Сумма двух точек вычисляется оператором `+`. Возвращается новый объект `sum_point`, который также является точкой на той же кривой, что и исходные точки. В конце, выводятся координаты результирующей точки на экран, рисуется красная точка на графике, представляющая результирующую точку сложения, и проверяется, является ли полученная точка бесконечно удаленной (x == None).

Результат работы программы

```
Сложение точек
Сумма двух точек (0, 1) и (3, 3): (3, 2)
```




```python
fig, ax = plt.subplots(1)
display(points, fig, ax)

# Точка для удвоения
print("Удвоение точки")
point3 = Point(int(input("Введите x = ")), int(input("Введите y = ")), a, b, p)

# Находим удвоение точки
double_point = point3.double()
print(f"Удвоение точки ({point3.x}, {point3.y}): ({double_point.x}, {double_point.y})")
ax.scatter(double_point.x, double_point.y, color = "r")
```

В этой части кода определен блок, который позволяет пользователю находить удвоенную точку на заданной кривой.

Создается объект `fig`, представляющий из себя одно изображение графика, и объект `ax`, который представляет собой область для рисования графика. Далее вызывается функция `display()`, которая отображает на графике все точки из списка `points`.

Далее пользователю предлагается ввести координаты точки `point3`, которая будет удваиваться, при помощи функций `input()`. Эти значения используются для создания объекта класса `Point` с именем `point3`.

Затем вызывается метод `double()` у объекта `point3`. Метод вычисляет координаты удвоенной точки при помощи формул, которые описывались ранее. В данной реализации возвращается новый объект `double_point`, которые также является точкой на кривой. В конце, выводятся координаты удвоенной точки на экран, рисуется красная точка на графике, представляющая точку удвоения.

Результат работы программы

```
Удвоение точки
Удвоение точки (3, 3): (0, 4)

<matplotlib.collections.PathCollection at 0x1bebcd6b0a0>
```