

# 椭圆曲线编程练习报告

姓名：蒋衲言 学号：2313546 班级：信息安全班

## 一、源码部分

### •Class of elliptic curve.h

```
//点类
class Point {
public:
    int x;
    int y;
    bool isInfinity = false;
};

//椭圆曲线类
class EllipticCurve {
public:
    //椭圆曲线  $E_p(a,b): y^2 \equiv x^3 + ax + b \pmod{p}$ 
    int p;
    int a;
    int b;
    EllipticCurve(int p, int a, int b) {
        this->p = p;
        this->a = a;
        this->b = b;
    }

    //判断是否为椭圆曲线
    bool isEllipticCurve();
    //判断给定的点是否在椭圆曲线上
    bool isOnEC(Point P);
    //计算  $P+Q$ 
    Point addPoint(Point P, Point Q);
    //使用“倍加-和”算法计算  $mP$  (其中  $m \leq \text{ord}(P)$ )
    Point DoubleAndAdd(int m, Point P);
    //计算点  $P$  的阶  $\text{ord}(P)$ 
    int ord(Point P);
    //计算椭圆曲线的阶  $\#E$ 
    int ordOfEC();
    //输出椭圆曲线上的所有点
    void printAllPoints();
};

//其他函数
//判断是否为大于 3 的质数
bool isPrime3(int a);
//求逆元  $a^{-1} \pmod{b}$ 
int modInverse(int a, int b);
```

### •Elliptic curve.cpp

```
#include<iostream>
#include"Class of elliptic curve.h"
using namespace std;

bool isPrime3(int a) {
    if (a <= 3) {
        return false;
    }
}
```

```

    bool judge = true;
    for (int i = 2; i * i <= a; i++) {
        if (a % i == 0) {
            judge = false;
            break;
        }
    }
    return judge;
}

int modInverse(int a, int b) {
    int mod = b;
    int s = 0;
    int s0 = 1, s1 = 0;
    int m = 0;
    while (a % b != 0) {
        s = s0 - (a / b) * s1;
        s0 = s1;
        s1 = s;
        m = a % b;
        a = b;
        b = m;
    }
    while (s < 0) {
        s += mod;
    }
    return s;
}

bool EllipticCurve::isEllipticCurve() {
    //p 为大于 3 的质数并且  $4a^3+27b^2$  不被 p 整除即为椭圆曲线
    return isPrime3(p) && (4 * a * a * a + 27 * b * b) % p != 0;
}

bool EllipticCurve::isOnEC(Point P) {
    return (P.isInfinity == true) || ((P.y * P.y - P.x * P.x * P.x - a * P.x - b) % p == 0);
}

Point EllipticCurve::addPoint(Point P, Point Q) {
    if (P.isInfinity == true) {
        return Q;
    } //O + Q = Q
    if (Q.isInfinity == true) {
        return P;
    } //P + O = P
    Point R;
    int k;
    if (P.x == Q.x) {
        if ((P.y + Q.y) % p == 0) {
            R.isInfinity = true;
            return R;
        } //P + Q = O
        k = ((3 * P.x * P.x + a) * modInverse(2 * P.y, p)) % p;
    }
    else {
        k = ((Q.y - P.y) * modInverse(Q.x - P.x, p)) % p;
    }
    R.x = (k * k - P.x - Q.x) % p;
    while (R.x < 0) {
        R.x += p;
    }
}

```

```

        R.y = (k * (P.x - R.x) - P.y) % p;
        while (R.y < 0) {
            R.y += p;
        }
        R.isInfinity = false;
        return R;
    }
    Point EllipticCurve::DoubleAndAdd(int m, Point P) {
        Point mP;
        mP.isInfinity = true;
        while (m > 0) {
            if (m % 2 == 1) {
                mP = addPoint(mP, P);
            }
            P = addPoint(P, P);
            m /= 2;
        }
        return mP;
    }
    int EllipticCurve::ord(Point P) {
        int ord = 0;
        for (int i = 1; i <= ordOfEC(); i++) {
            if (DoubleAndAdd(i, P).isInfinity) {
                ord = i;
                break;
            }
        }
        return ord;
    }
    int EllipticCurve::ordOfEC() {
        int ord = 1; // 初始化为 1, 因为要算上无穷远点 O
        for (int x = 0; x < p; x++) {
            for (int y = 0; y < p; y++) {
                if ((y * y - x * x * x - a * x - b) % p == 0) {
                    ord++;
                }
            }
        }
        return ord;
    }
    void EllipticCurve::printAllPoints() {
        int count = 0;
        for (int x = 0; x < p; x++) {
            for (int y = 0; y < p; y++) {
                if ((y * y - x * x * x - a * x - b) % p == 0) {
                    count++;
                    cout << "(" << x << "," << y << ")" << ",";
                    if (count % 4 == 0) {
                        cout << endl;
                    }
                }
            }
        }
        cout << "O" << endl;
    }
}

```

#### ●Main function.cpp

```
#include<iostream>
```

```

#include "Class of elliptic curve.h"
using namespace std;

int main() {
    bool whetherLoop = true;

    //1.给定参数 p,a,b, 判断 E_p(a,b)是否为椭圆曲线
    cout << "1.给定参数 p,a,b, 判断 E_p(a,b)是否为椭圆曲线" << endl;
    int p, a, b;
    while (whetherLoop) {
        cout << "请输入椭圆曲线 E_p(a,b):y^2=x^3+ax+b(mod p)的参数: " << endl;
        cout << "p=";
        cin >> p;
        cout << "a=";
        cin >> a;
        cout << "b=";
        cin >> b;
        EllipticCurve e(p, a, b);
        if (!e.isEllipticCurve()) {
            cout << "E_" << p << "(" << a << "," << b << ")"
                << "不是椭圆曲线, 请重新输入! " << endl;
        }
        else {
            cout << "E_" << p << "(" << a << "," << b << ")" << "是椭圆曲线。" << endl;
            whetherLoop = false;
        }
    }
    EllipticCurve e(p, a, b);
    whetherLoop = true;

    //2.判断给定的点 P,Q 是否在椭圆曲线 E_p(a,b)上
    cout << "2.判断给定的点 P,Q 是否在椭圆曲线 E_p(a,b)上" << endl;
    Point P, Q;
    while (whetherLoop) {
        cout << "请输入点 P 的坐标: " << endl;
        cout << "横坐标 x1=";
        cin >> P.x;
        cout << "纵坐标 y1=";
        cin >> P.y;
        if (!e.isOnEC(P)) {
            cout << "点 P(" << P.x << "," << P.y << ")" << "不在椭圆曲线" << "E_" << p
                << "(" << a << "," << b << ")" << "上, 请重新输入! " << endl;
        }
        else {
            cout << "点 P(" << P.x << "," << P.y << ")" << "在椭圆曲线" << "E_" << p
                << "(" << a << "," << b << ")" << "上。" << endl;
            whetherLoop = false;
        }
    }
    whetherLoop = true;
    while (whetherLoop) {
        cout << "请输入点 Q 的坐标: " << endl;
        cout << "横坐标 x2=";
        cin >> Q.x;
        cout << "纵坐标 y2=";
        cin >> Q.y;
        if (!e.isOnEC(Q)) {
            cout << "点 Q(" << Q.x << "," << Q.y << ")" << "不在椭圆曲线" << "E_" << p

```

```

        << "(" << a << ", " << b << ")" << "上，请重新输入！" << endl;
    }
    else {
        cout << "点 Q(" << Q.x << ", " << Q.y << ")" << "在椭圆曲线" << "E_" << p
            << "(" << a << ", " << b << ")" << "上。" << endl;
        whetherLoop = false;
    }
}
whetherLoop = true;

//3.对在椭圆曲线 E_p(a,b)上的两点 P,Q，计算 P+Q
cout << "3.对在椭圆曲线 E_p(a,b)上的两点 P,Q，计算 P+Q" << endl;
cout << "P(" << P.x << ", " << P.y << ") + Q(" << Q.x << ", " << Q.y << ") = ";
if (e.addPoint(P, Q).isInfinity == true) {
    cout << "O" << endl;
}
else {
    cout << "(" << e.addPoint(P, Q).x << ", " << e.addPoint(P, Q).y << ")" << endl;
}

//4.对在椭圆曲线 E_p(a,b)上的点 P，使用“倍加-和”算法计算 mP
cout << "4.对在椭圆曲线 E_p(a,b)上的点 P，使用“倍加-和”算法计算 mP" << endl;
int m = 0;
cout << "请输入 m 的值：m=";
cin >> m;
cout << m << "P=";
if (e.DoubleAndAdd(m, P).isInfinity == true) {
    cout << "O" << endl;
}
else {
    cout << "(" << e.DoubleAndAdd(m, P).x << ", "
        << e.DoubleAndAdd(m, P).y << ")" << endl;
}

//5.对在椭圆曲线 E_p(a,b)上的点 P，计算阶 ord(P)
cout << "5.对在椭圆曲线 E_p(a,b)上的点 P，计算阶 ord(P)" << endl;
cout << "点 P 的阶 ord(P)=" << e.ord(P) << endl;

//6.对于椭圆曲线 E_p(a,b)，计算阶 #E
cout << "6.对于椭圆曲线 E_p(a,b)，计算阶 #E" << endl;
cout << "椭圆曲线" << "E_" << p << "(" << a << ", " << b << ")"
    << "的阶 #E=" << e.ordOfEC() << endl;

//7.对于椭圆曲线 E_p(a,b)，计算所有点
cout << "7.对于椭圆曲线 E_p(a,b)，计算所有点" << endl;
cout << "椭圆曲线" << "E_" << p << "(" << a << ", " << b << ")"
    << "上所有点为" << endl;
e.printAllPoints();

//8.其他功能的进一步扩展.....

system("pause");
}

```

## 二、说明部分

### 1. 头文件 Class of elliptic curve.h

Point 类：用于表示椭圆曲线上的点。每个点有 x 和 y 坐标，以及一个 isInfinity 布尔值，

用于表示该点是否为无穷远点。

**EllipticCurve 类：**用于表示椭圆曲线  $E_p(a,b)$ ，其中  $p$  是一个质数， $a$  和  $b$  是满足特定条件的整数 ( $4a^3+27b^2 \not\equiv 0 \pmod{p}$ )。该类包含多个成员函数，用于：

- 检查给定的参数是否构成一个椭圆曲线。
- 检查一个点是否在椭圆曲线上。
- 计算两个点的加法。
- 使用“倍加-和”算法计算  $mP$ 。
- 计算点  $P$  的阶  $\text{ord}(P)$  (即最小的正整数  $n$ ，使得  $nP$  是无穷远点)。
- 计算椭圆曲线的阶  $\#E$  (即椭圆曲线上点的总数，包括无穷远点)。
- 输出椭圆曲线上的所有点。

## 2. 源文件 Elliptic curve.cpp

- 实现了 `isPrime3` 函数，用于检查一个数是否为大于 3 的质数。
- 实现了 `modInverse` 函数，用于计算逆元 (即给定  $a$  和  $b$ ，找到  $x$  使得  $ax \equiv 1 \pmod{b}$ )。
- 实现了 `EllipticCurve` 类中的成员函数，这些函数基于椭圆曲线的数学性质进行计算。

## 3. 主函数 Main function.cpp

提供了用户交互界面，允许用户输入椭圆曲线的参数  $p,a,b$ ，以及点  $P,Q$  的坐标和倍数  $m$ 。通过调用 `EllipticCurve` 类中的成员函数，执行以下操作：

- 检查给定的参数是否构成一个椭圆曲线。
- 检查点  $P$  和  $Q$  是否在椭圆曲线上。
- 计算点  $P$  和  $Q$  的加法结果。
- 使用“倍加-和”算法计算  $mP$ 。
- 计算点  $P$  的阶。
- 计算椭圆曲线的阶。
- 输出椭圆曲线上的所有点。

## 三、运行示例

```
Microsoft Visual Studio 调试 × + ▾

请输入点P的坐标：
横坐标x1=1
纵坐标y1=2
点P(1,2)不在椭圆曲线E_19(3, 7)上，请重新输入！
请输入点P的坐标：
横坐标x1=1
纵坐标y1=7
点P(1,7)在椭圆曲线E_19(3, 7)上。
请输入点Q的坐标：
横坐标x2=3
纵坐标y2=9
点Q(3,9)在椭圆曲线E_19(3, 7)上。
3.对椭圆曲线E_p(a,b)上的两点P,Q，计算P+Q
P(1,7)+Q(3,9) = (16,16)
4.对椭圆曲线E_p(a,b)上的点P，使用“倍加-和”算法计算mP
请输入m的值：m=7
7P=(15,11)
5.对椭圆曲线E_p(a,b)上的点P，计算阶ord(P)
点P的阶ord(P)=11
6.对于椭圆曲线E_p(a,b)，计算阶#E
椭圆曲线E_19(3, 7)的阶#E=22
```

7.对于椭圆曲线 $E_p(a,b)$ , 计算所有点

椭圆曲线 $E_{19}(3, 7)$ 上所有点为

$(0,8), (0,11), (1,7), (1,12),$   
 $(3,9), (3,10), (4,8), (4,11),$   
 $(8,7), (8,12), (10,7), (10,12),$   
 $(12,2), (12,17), (13,1), (13,18),$   
 $(14,0), (15,8), (15,11), (16,3),$   
 $(16,16), 0$

请按任意键继续 . . .

E:\C++Code\Elliptic Curve\x64\Debug\Elliptic Curve.exe

按任意键关闭此窗口 . . .