# 第5次编程练习报告

姓名：曹瑜 学号：2212794 班级：密码科学与技术

##### **编程练习1——实现基本的Zp上的椭圆曲线Ep（a，b）的计算**

* **源码部分：**

#include <iostream>

#include <utility>

#include<vector>

using namespace std;

int gcd(int a, int b) {

if (a < b) {

int temp = a;

a = b;

b = temp;

}

if (b == 0) { return a; }

else {

return gcd(b, a % b);

}

}

int exgcd(int a, int b, int& x, int& y) {

if (b == 0) {

x = 1;

y = 0;

return a;

}

int x1, y1;

int d = exgcd(b, a % b, x1, y1);

x = y1;

y = x1 - (a / b) \* y1;

return d;

}

//求逆元

int modInverse(int a, int m) {

int x, y;

int d = exgcd(a, m, x, y);

if (d != 1) {

return -1; // 若 a 和 m 不互质，则无逆元

}

return (x % m + m) % m;

}

// 点加运算 (不同点)

pair<int, int> pointAdd(int a, int b, int p, pair<int, int> P, pair<int, int> Q) {

if (P.first == -1 && P.second == -1) return Q; // P 是无穷远点

if (Q.first == -1 && Q.second == -1) return P; // Q 是无穷远点

if (P.first == Q.first) return { -1, -1 }; // 竖直线，P + Q = 无穷远点

int denominator = Q.first - P.first;

int inv = modInverse(denominator, p);

if (inv == -1) return { -1, -1 }; // 无逆元，说明运算不合法

int s = (Q.second - P.second) \* inv % p;

if (s < 0) s += p;

int xr = (s \* s - P.first - Q.first) % p;

if (xr < 0) xr += p;

int yr = (s \* (P.first - xr) - P.second) % p;

if (yr < 0) yr += p;

return { xr, yr };

}

// 倍点运算

pair<int, int> pointDouble(int a, int b, int p, pair<int, int> P) {

if (P.first == -1 && P.second == -1) return P; // P 是无穷远点

if (P.second == 0) return { -1, -1 }; // P + P = 无穷远点

int denominator = 2 \* P.second;

int inv = modInverse(denominator, p);

if (inv == -1) return { -1, -1 }; // 无逆元，说明运算不合法

int s = (3 \* P.first \* P.first + a) \* inv % p;

if (s < 0) s += p;

int xr = (s \* s - 2 \* P.first) % p;

if (xr < 0) xr += p;

int yr = (s \* (P.first - xr) - P.second) % p;

if (yr < 0) yr += p;

return { xr, yr };

}

// 计算点的 n 倍 (倍加-和算法)

pair<int, int> pointMultiply(int a, int b, int p, int n, pair<int, int> P) {

pair<int, int> Q = { -1, -1 }; // 初始化为无穷远点

pair<int, int> R = P;

while (n > 0) {

if (n % 2 == 1) {

Q = pointAdd(a, b, p, Q, R);

}

R = pointDouble(a, b, p, R);

n /= 2;

}

return Q;

}

//判断是否是椭圆曲线

void is\_Ellipticcurve(int a,int b,int p) {

int result = 4 \* a \* a \* a + 27 \* b \* b;

result %= p;

if (result == 0) {

cout<<"E\_"<<p<<"("<<a<<","<<b<<")"<<"is not elliptic curve "<<endl;

}

else {

cout<< "E\_" << p << "(" << a << "," << b << ")" << "is elliptic curve "<<endl;

}

cout << endl;

}

//判断点是否在椭圆曲线上

void is\_it(int a, int b, int p, int x, int y) {

int R,L;

R = (x \* x \* x + a \* x + b) % p;

L = (y \* y) % p;

if (R == L) {

cout << "(" << x << "," << y << ") is on " << "E\_" << p << "(" << a << "," << b << ")"<<endl;

}

else {

cout<< "(" << x << "," << y << ") is not on " << "E\_" << p << "(" << a << "," << b<< ")"<<endl;

}

cout << endl;

}

//计算点的阶

int ord(int a,int b,int p,pair<int, int> P) {

int n = 1;

while (pointMultiply(a, b, p, n, P).first != -1 && pointMultiply(a, b, p, n, P).second != -1) {

n++;

}

return n;

}

// 计算勒让德符号

int legendreSymbol(int a, int p) {

int ls = static\_cast<int>(pow(a, (p - 1) / 2)) % p;

return ls == p - 1 ? -1 : ls;

}

// 计算椭圆曲线上的点

pair<vector<pair<int, int>>, int> ellipticCurvePoints(int a, int b, int p) {

vector<pair<int, int>> points;

for (int x = 0; x < p; x++) {

int lhs = (x \* x \* x + a \* x + b) % p;

if (lhs < 0) lhs += p; // 确保 lhs 为非负数

int rhs = legendreSymbol(lhs, p);

if (rhs == 1) {

for (int y = 0; y < p; y++) {

if ((y \* y) % p == lhs) {

points.emplace\_back(x, y);

}

}

}

}

return { points, static\_cast<int>(points.size()) };

}

int main() {

int a, b, p, n ;

//椭圆曲线判断

cout << "please cin a,b,p:";

cin >> a >> b >> p;

is\_Ellipticcurve(a, b, p);

//判断点是否在椭圆曲线上

int x1, y1, x2, y2;

cout << "please cin (x,y):";

cin >> x1 >> y1;

is\_it(a, b, p, x1, y1);

cout << "please cin (x,y):";

cin >> x2 >> y2;

is\_it(a, b, p, x2, y2);

//计算P+Q：

int x3, y3, x4, y4;

cout << "P: ";

cin >> x3 >> y3;

cout << "Q: ";

cin >> x4 >> y4;

cout << "P+Q = ";

pair <int, int> Result= pointAdd(a, b, p, { x3,y3 }, { x4,y4 });

cout << "(" << Result.first << "," << Result.second << ")"<<endl<<endl;

//计算2P：

int x5, y5;

cout << "P: ";

cin >> x5 >> y5;

cout << "2P = (" << pointDouble(a, b, p, { x5,y5 }).first<<","<< pointDouble(a, b, p, {x5,y5}).second <<")"<< endl << endl;

//计算nP：

cout << "please cin n,P(x),P(y):";

int x6, y6;

cin>>n >> x6 >> y6;

cout << n << "P= (" << pointMultiply(a, b, p, n, { x6,y6 }).first << "," << pointMultiply(a, b, p, n, { x6,y6 }).second << ")" << endl << endl;

//计算ord（p）

cout << "please cin P:";

int x7, y7;

cin >> x7 >> y7;

cout << "ord(P)=" << ord(a, b, p, { x7,y7 }) << endl << endl;

//计算#E

int pointCount,c,d,P;

cout << "please cin a,b,p: ";

cin >> c >> d >> P;

// 输出椭圆曲线上的所有点

pair<vector<pair<int, int>>, int> curvePoints = ellipticCurvePoints(a, b, p);

int numPoints = (curvePoints.second+1);

cout << "#E\_(" << c << "," << d << ")= " << numPoints<<endl;

for (const auto& point : curvePoints.first) {

cout << "(" << point.first << ", " << point.second << ")" << endl;

}

cout << " O";

return 0;

}

* **说明部分：**//主要说明实现的一些基本原理等

判断是否是椭圆曲线部分：计4a3+27b2 ≡ 0（mod p）

判断点是否在椭圆曲线上部分：将x和y的值代入，分别计算同余号左右两边的值，看是否相等，若同余式成立则该点确实在该椭圆曲线上；

点加运算部分：先判断两点中是否有无穷远点，若有则结果为另一点的值（P+O=P）；再判断两点是否为x值相同的对称点，若为则结果为O（P+(-P)=O）；再根据公式计算点加结果；

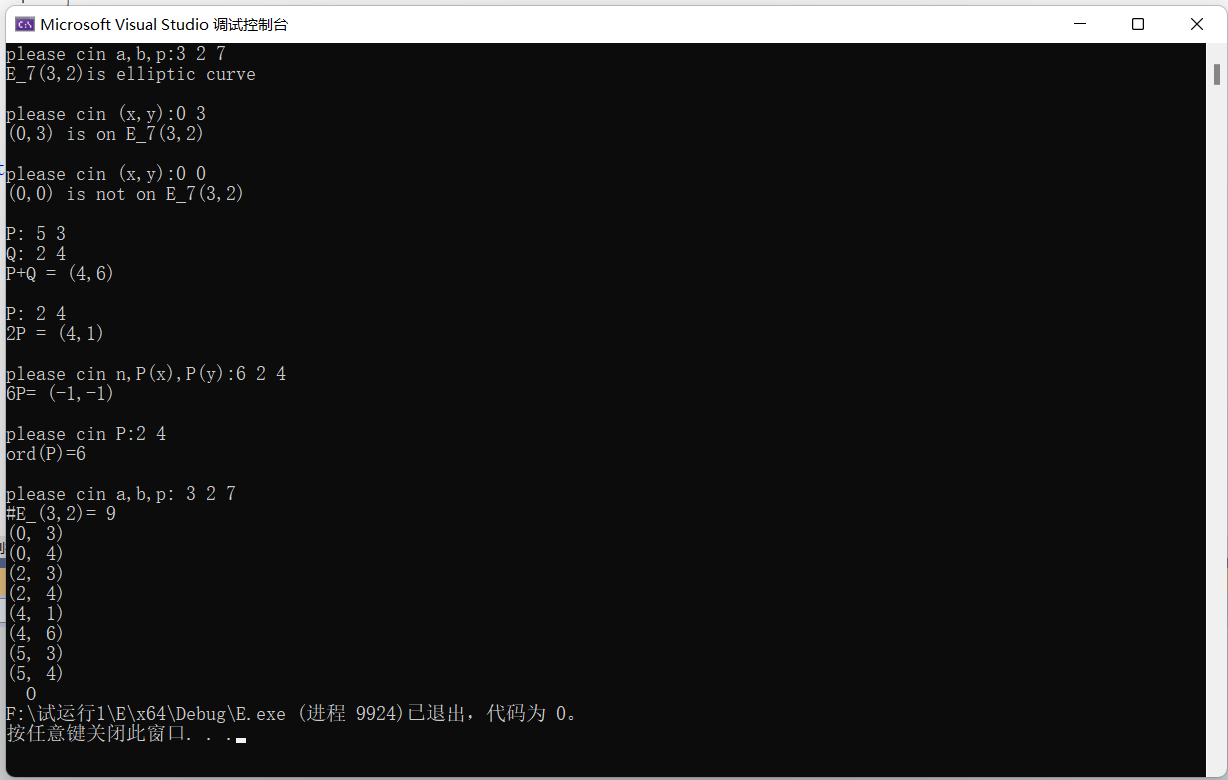
点倍加运算部分：先判断该点是否为无穷远点或x轴上的点，若为则结果为O，再根据公式计算倍加结果；

倍加-和算法：通过二进制表示n，将n分解为二进制数的每一位，然后根据每一位对点进行加法或者直接对点进行双倍运算来实现；

计算点的阶：从n=2开始依次计算nP的值，当计算到nP=O时返回此时的n值即为ord（P）；

计算椭圆曲线Ep（a，b）的阶/其上所有点：遍历x从0到p-1的值，，，，，，最后返回points以及点的数量。

* **运行示例：**//截图



* **其他：**//用于回答可能预留的问题