КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет комп'ютерних наук та кібернетики

Кафедра інтелектуальних програмних систем

Лабораторна робота №4

з предмету «Математичні основи захисту інформації»

Варіант №13

Підготував:

Грищенко Юрій, ІПС-42

Київ – 2022

Завдання 1.

Визначити, чи існують такі числа p,q, що число n є різницею їх квадратів. Які при цьому числа r та s, що n є їх добутком?

```
1. n = 14873
   Відповідь:
   а) не існують
   б) p = 123
      q = 16
      r = 139
      s = 107
2. n = 284350889
   Відповідь:
   а) не існують
   б) p = 16875
      q = 644
      r = 17519
      s = 16231
```

Теорія

Метод факторизації Ферма — алгоритм факторизації (розклад на множники) непарного цілого числа **n**, представлений П'єром Ферма у 1643 році.
Метод заснований на пошуку таких цілих чисел x і y, які задовольняють відношення $x^2-y^2=n$, що веде до розкладу n = (x-y)(x+y).

Для розкладання на множники непарного числа шукається пара чисел x, y таких, що $x^2-y^2=n$, або (x-y)(x+y)=n. При цьому числа (x-y) і (x+y) є множниками , можливо, тривіальними (тобто одне з них дорівнює 1, а інше — n.)

У нетривіальному випадку, рівність $x^2-y^2=n$ рівносильно $x^2-n=y^2$, тобто того, що x^2-n є квадратом.

Пошук квадрата такого виду починається з $x=\lceil\sqrt{n}\rceil$ — найменшого числа, при якому різниця невід'ємна.

Для кожного значення k $\in$ N починаючи з 1, обчислюють $(\lceil\sqrt{n}\rceil+k)^2-n$ і перевіряють, чи не є це число точним квадратом. Якщо не є, то k збільшують на одиницю і переходять на наступну ітерацію.

Якщо $(\lceil\sqrt{n}\rceil+k)^2-n$ є точним квадратом, то отримано розкладання:

$n=x^2-y^2=(x+y)(x-y)=a*b$ в якому $x=\lceil\sqrt{n}\rceil+k$

Якщо воно є тривіальним і єдиним, то n — просте.

На практиці значення виразу на k-ому кроці вираховується з врахуванням значення на (k+1)-ому кроці:

$(s+1)^2 - n = s^2 + 2s + 1 - n$ де $s = \lceil \sqrt{n} \rceil + k$

Код програми

```
private static class FermaFactorizationResult {
    //n = p^2 - q^2
    public final int p;
    public final int q;
    //n = r * s
    public final int r;
    public final int s;

    FermaFactorizationResult(int p, int q, int r, int s) {
        this.p = p;
        this.q = q;
        this.r = r;
        this.s = s;
    }
}

public static FermaFactorizationResult fermaFactorizeStep(int n) {
    double s = Math.ceil(Math.sqrt(n));

    for (int k = 0; k < n; k++) {
        double y = (s + k) * (s + k) - n;

        double ySqrt = Math.sqrt(y);
        int q = (int) ySqrt;
        if (ySqrt == q) {
            int p = (int) (s + k);
            return new FermaFactorizationResult(
                    p,
                    q,
                    p + q,
                    p - q
            );
        }
    }
    throw new ArithmeticException("Not a difference of squares (" + n + " is an even 
number?)");
}
```

```
for (int n : new int[]{14873, 284350889}) {
    FermaFactorizationResult result = fermaFactorizeStep(n);
    System.out.printf("%d = %d^2 - %d^2 = %d * %d\n", n, result.p, result.q, result.r, 
result.s);
}
```

Приклади

```
14873 = 123^2 - 16^2 = 139 * 107
284350889 = 16875^2 - 644^2 = 17519 * 16231
```

Завдання 2 (13).

Знайти кількість натуральних чисел, менших n і взаємно простих з n, якщо
а) n = 3560;
б) n = 4520;
в) n = 116424;
г) n = 1002001;
д) n = 1294700;
е) n = 1294699.

Теорія

Функція Ойлера φ(n) визначається для всіх натуральних чисел n, значенням якої є кількість натуральних чисел, менших від n і взаємно простих з n. Для n = 1 покладають φ(1) = 1.

Для невеликих значень n значення φ(n) можна знайти простим підрахунком.

Для обчислення значення φ(n) для довільного n існує формула, за якою ці значення знаходяться.

$$\varphi(n) = \frac{n}{p_1 p_2 \ldots p_k}(p_1 - 1)(p_2 - 1)\ldots(p_k - 1) = n(1 - \frac{1}{p_1})(1 - \frac{1}{p_2})\ldots(1 - \frac{1}{p_k}),$$

Для знаходження дільників числа n (p_1, p_2, …) скористаємося рекурсивним алгоритмом для методу факторизації Ферма з минулого завдання.

Код програми

```
    public static void fermaFactorizeRecursive(int n, List<Integer> factors) {
        while (n % 2 == 0) {
            n /= 2;
            factors.add(2);
        }
        FermaFactorizationResult result = fermaFactorizeStep(n);
//        System.out.printf("n = %d, r = %d, s = %d\n", n, result.r, result.s);
        if (result.s == 1) {
            factors.add(result.r);
        } else {
            fermaFactorizeRecursive(result.r, factors);
            fermaFactorizeRecursive(result.s, factors);
        }
    }

    public static int euler(int n) {
        ArrayList<Integer> factorsList = new ArrayList<>();
        fermaFactorizeRecursive(n, factorsList);

        HashSet<Integer> factorsSet = new HashSet<>(factorsList);
//        System.out.println(factorsSet);

        double result = n;
        for (int factor : factorsSet) {
            result *= 1. - 1. / factor;
        }
```

```
        return (int) result;
    }
```

```
for (int n : new int[]{3560, 4520, 116424, 1002001, 1294700, 1294699}) {
    System.out.printf("phi(%d) = %d\n", n, euler(n));
}
```

Результати

```
phi(3560) = 1408
phi(4520) = 1792
phi(116424) = 30240
phi(1002001) = 720720
phi(1294700) = 466400
phi(1294699) = 1109736
```

Завдання 2 (18).

Дано φ(n) = 1792 і n = $2^x 5^y 113^z$. Знайти n.

Теорія

Скористаємося формулою

$$\varphi(n) = \frac{n}{p_1 p_2 \ldots p_k}(p_1 - 1)(p_2 - 1)\ldots(p_k - 1) = n(1 - \frac{1}{p_1})(1 - \frac{1}{p_2})\ldots(1 - \frac{1}{p_k}),$$

Тоді

n = φ(n) / ((1-1/2)(1-1/5)(1-1/113))

Код програми

```
double task18 = 1792 / (1 - 1./2) / (1 - 1./5) / (1 - 1./113);
ArrayList<Integer> factors = new ArrayList<>();
fermaFactorizeRecursive((int)task18, factors);

System.out.println(task18);
System.out.println(factors);
```

Результати

```
4520.0
[2, 2, 2, 113, 5]
```

Завдання 6 (25)

Перевірити справедливість конгруенцій: $5^{\varphi(26)} \equiv 1 (\mathrm{mod}\ 26)$, $2^{\varphi(45)} \equiv 1 (\mathrm{mod}\ 45)$, $3^{\varphi(40)} \equiv 1\ (\mathrm{mod}\ 40)$.

Теорія

Теорема Ойлера: Якщо m – натуральне число і a – таке ціле число, що НСД(a, m) = 1, то $a^{\varphi(m)} \equiv 1\ (\mathrm{mod}\ m)$, де φ(m) – функція Ойлера.

Дійсно, 5 та 26, 2 та 45, 3 та 40 — взаємнопрості пари чисел, отже конгруенція справедлива.

Можна ще перевірити за допомогою коду.

Код програми

```
for (int[] tuple : new int[][]{{5, 26}, {2, 45}, {3, 40}}) {
    int result = (int)Math.pow(tuple[0], euler(tuple[1])) % tuple[1];
    System.out.printf("%d^phi(%d) = %d (mod %d)\n", tuple[0], tuple[1], result, tuple[1]);
}
```

Результати

```
5^phi(26) = 1 (mod 26)
2^phi(45) = 1 (mod 45)
3^phi(40) = 1 (mod 40)
```

Перелік літературних джерел

1. Кривий С.Л. Конспект лекцій «Математичні основи захисту інформації»
2. https://uk.wikipedia.org/wiki/Метод_факторизації_Ферма