Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»
(СибГУТИ)

Кафедра прикладной математики и кибернетики

РАСЧЕТНО-ГРАФИЧЕСКОЕ ЗАДАНИЕ
по дисциплине «Защита информации»

На тему: «Доказательство с нулевым знанием для задачи Гамильтонов цикл»

Выполнил:
студент гр. ИП-613

Чистов Д.А.

Работу проверил:
ассистент кафедры ПМиК

Петухова Я. В.

Новосибирск 2019 г.

Постановка задачи

Необходимо написать программу, реализующую протокол доказательства с нулевым знанием для задачи «Гамильтонов цикл». Нахождение Гамильтонова цикла является NP-полной задачей и не имеет быстрых методов для решения, поэтому генерация графов с Гамильтоновым циклом в решение этой задачи не входит. Необходимо информацию о графах считывать из файла.

В файле описание графа будет определяться следующим образом:

1) в первой строке файла содержатся два числа $0 < n < 1001$ и $0 < m <= n^2$, количество вершин графа и количество рѐбер соответственно;

2) в последующих m строках содержится информация о рѐбрах графа, каждое из которых описывается с помощью двух чисел (номера вершин, соединяемых этим ребром);

3) в зависимости от варианта указывается необходимая дополнительная информация: в первом варианте перечисляются цвета вершин графа; во втором варианте указывается последовательность вершин, задающая гамильтонов цикл (этот пункт можно вынести в отдельный файл).

Алгоритм решения задачи

Важный элемент, что зашифрованные в этой системе сообщения может расшифровать только Алиса и больше никто. Протокол доказательства состоит из следующих четырех шагов (пояснения будут даны ниже).

Шаг 1. Алиса строит граф H, являющийся копией исходного графа G, где у всех вершин новые, случайно выбранные номера. На языке теории графов говорят, что H изоморфен G. Иными словами, H получается путем некоторой перестановки вершин в графе G (с сохранением связей между вершинами). Алиса кодирует матрицу H, приписывая к первоначально содержащимся в ней нулям и единицам случайные числа r_{ij} по схеме $H'_{ij} = r_{ij}||H_{ij}$. Затем она шифрует элементы матрицы H', получая зашифрованную матрицу F , $F_{ij} = H'd_{ji} \bmod N$. Матрицу F Алиса передает Бобу.

Шаг 2. Боб, получив зашифрованный граф F , задает Алисе один из двух вопросов.

1. Каков гамильтонов цикл для графа H?
2. Действительно ли граф H изоморфен G?

Шаг 3. Алиса отвечает на соответствующий вопрос Боба.

1. Она расшифровывает в F ребра, образующие гамильтонов цикл.

   Боб не может применить этот цикл к графу G, так как H1 и F изоморфны к нему.
2. Она расшифровывает F полностью(фактическипередает Бобу граф H') и предъявляет перестановки, с помощью которых граф H был получен из графа G.

   Бобу может быть известен цикл полученный при задании первого вопроса, но он не валиден, т.к. графы H, H' и F были перестроены и имеют другие перестановки.

   То есть Боб не может построить граф с гамильтоновым циклом если он за один раз получает только один ответ.

Шаг 4. Получив ответ, Боб проверяет правильность расшифровки путем повторного шифрования и сравнения с F и убеждается либо в том, что показанные ребра действительно образуют гамильтонов цикл, либо в том, что предъявленные перестановки действительно переводят граф G в граф H.

Весь протокол повторяется t раз.

Приложение

Пример выполнения программы

```
N:840547360863827 D:853366465
G:
0 1 0 0 1
1 0 1 1 0
0 1 0 1 0
0 1 1 0 1
1 0 0 1 0
H:
0 1 1 1 0
1 0 0 0 1
1 0 0 1 0
1 0 1 0 1
0 1 0 1 0
H1:
349870 325041 573231 745841 402110
246241 188630 100460 50920 965981
322411 703250 918170 491311 631330
660231 714370 28391 426330 72261
808940 47251 857020 242741 737320
F:
422635158793977 274102023861856 722410826325658 631628543686807 143724038870686
502470225698306 758023055182989 741470920222259 285943329741121 563062941230258
515819021733062 132376606519182 159675106907892 384342857709302 640596874656644
488306177281007 454066027824536 707879622805557 148375396645807 280413200826015
715510328865002 37989035057629 333374011593735 68854494226878 29162381105498
```

Рис1.Выполнение Шаг 1

```
F:
525094148598986444 238239007118924839 581958522840968679 477196129132417120 182092837628434362
186382465910993184 452816225369854718 109067335531169480 398888641981785013 122221679258677374
75423259816812683 300132698430753479 103198252988704172 353376387645846801 456820902448882168
94268299231156650 464132212956759233 50098502875871280 433424150479988346 214527327149148678
71037678781332378 224292054862092817 351545922058981787 381719257817982316 418920798046281068
2 1 :194331 186382465910993184
1 3 :315231 581958522840968679
3 4 :107281 353376387645846801
4 5 :450101 214527327149148678
5 2 :835551 224292054862092817
```

Рис2. Шаг 3 проверка Гамильтонов

```
F1:
0 1 0 0 1
1 0 1 1 0
0 1 0 1 0
0 1 1 0 1
1 0 0 1 0
```

Рис3. Шаг 3 проверка расшифрованного графа F с перестановками

Листинг

Main.java

```java
package com.lab1;

import java.io.IOException;


public class Main {
    public static void main(String[] args) throws IOException {
        //RGZ
        Gamelt.generation();
    }

}
```

Gamelt.java

```java
package com.lab1;

import java.io.File;
```

```java
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.Random;
import java.util.Scanner;

import static com.lab1.Criptogr.check;
import static com.lab1.Criptogr.exponentiation;

public class Gamelt {

    public static void generation() throws FileNotFoundException {

        Random random = new Random();
        int temp, temp1;
        int n, m;
        long P, Q;
        do {
            P = random.nextInt(999999999) + 2;
        } while (check(P) == false);
        do {
            Q = random.nextInt(999999999) + 2;
        } while (check(Q) == false);

        long f = (Q - 1) * (P - 1);
        long N = P * Q;

        long d, c;
        long[] resultEcklid;

        while(true) {
            if (f < 1000000000) {
                d = random.nextInt((int)f);
            } else {
                d = random.nextInt(1000000000);
            }
            resultEcklid = Criptogr.evklid(d, f);
            if (resultEcklid[0] == 1) {
                break;
            }
        }
        if (resultEcklid[1] < 0) {
            resultEcklid[1] += f;
        }
        c = resultEcklid[1];
        Scanner in = new Scanner(new File("resource\\graf.txt"));
        temp = in.nextInt();
        if (!(temp > 0 && temp < 1001)) {
            return;
        }
        n = temp;
        temp = in.nextInt();
        if (!(temp > 0 && temp <= (n * n))) {
            return;
        }
        m = temp;

        int[][] G = new int[n][n];
        for (int i = 0; i < m; i++) {
            temp = in.nextInt() - 1;
            temp1 = in.nextInt() - 1;
            G[temp][temp1] = 1;
            G[temp1][temp] = 1;
```

```
        }

        ArrayList<Integer> Gam = new ArrayList<>();
        ArrayList<Integer> Gam1 = new ArrayList<>();
        while (in.hasNextInt()) {
            temp = in.nextInt();
            Gam.add(temp);
            Gam1.add(temp);
        }

        System.out.println("N:" + N + " D:" + d );
        System.out.println("G:" );
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                System.out.print(G[i][j] + " ");
            }
            System.out.println();
        }

        int[][] H = G;
        int pp = random.nextInt(n - 1) + 1;
        for (int i = 0; i < n; i++) {
            temp = H[i][0];
            H[i][0] = H[i][pp];
            H[i][pp] = temp;
        }

        for (int i = 0; i < n; i++) {
            temp = H[0][i];
            H[0][i] = H[pp][i];
            H[pp][i] = temp;
        }

        System.out.println("H:" );
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                System.out.print(H[i][j] + " ");
            }
            System.out.println();
        }

        try {
            temp = Gam.get(pp);
            Gam.set(pp, Gam.get(0));
            Gam.set(0, temp);
            Gam.set(Gam.size() - 1, temp);
        } catch (Exception e) {
            return;
        }

        int[][] H1 = H;
        String str;
        int r;
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                r = random.nextInt(100000);
                str = r + Integer.toString(H[i][j]);
                H1[i][j] = Integer.valueOf(str);
            }
        }

        System.out.println("H1: ");
```

```
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                System.out.print(H1[i][j] + " ");
            }
            System.out.println();
        }

        long[][] F =  new long[n][n];

        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                F[i][j] = exponentiation(H1[i][j], d, N).longValue();
            }
        }

        System.out.println("F: ");
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                System.out.print(F[i][j] + " ");
            }
            System.out.println();
        }

        //1 вопрос , ответ
        for (int i = 0; i < Gam.size() - 1; i++) {
            System.out.print(((Gam1.get(i) - 1) + 1) + " " + ((Gam1.get(i + 1) - 1) +
1) + " :");
            System.out.print(exponentiation(F[Gam.get(i) - 1][Gam.get(i + 1) - 1], c,
N).longValue() + " ");
            System.out.println(exponentiation(H1[Gam.get(i) - 1][Gam.get(i + 1) - 1],
d, N));
        }

        long[][] F1 = F;
        System.out.println("F1: ");
        //2 вопрос ответ
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                F1[i][j] = exponentiation(F1[i][j], c, N).longValue();
            }
        }

        for (int i = 0; i < n; i++) {
            temp = (int)F1[0][i];
            F1[0][i] = F1[pp][i];
            F1[pp][i] = temp;
        }

        for (int i = 0; i < n; i++) {
            temp = (int)F1[i][0];
            F1[i][0] = F1[i][pp];
            F1[i][pp] = temp;
        }

        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                if (F1[i][j] % 2 == 0) {
                    F1[i][j] = 0;
                } else {
                    F1[i][j] = 1;
                }
                System.out.print(F1[i][j] + " ");
```

```
            }
            System.out.println();
        }
    }
}
```