Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики»

(СибГУТИ)

Кафедра прикладной математики и кибернетики

РАСЧЕТНО-ГРАФИЧЕСКАЯ РАБОТА

по дисциплине «Защита информации»

Вариант 2

Выполнил:

студент гр. ИП-712

Алексеев С.В.

Работу проверил:

Ассистент кафедры ПМиК

Петухова Я.В.

Новосибирск 2020 г.

Задание.

Необходимо написать программу, реализующую протокол доказательства с нулевым знанием для задачи “Гамильтонов цикл”. в первой строке файла содержатся два числа n < 1001 и m <= n^2, количество вершин графа и количесвто рёбер соответсвенно. В последующих строках содержится информация о рёбрах и гамильтонов цикл.

Теория.

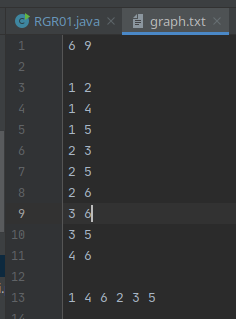
За основу возьмем криптографический протокол “Доказательство с нулевым знанием”.

Задача Алисы будет заключаться в том, чтобы передать Бобу «зашифрованный» граф и доказать, что данный граф является Гамильтоновым т.е имеет Гамильтонов цикл – путь проходит через каждую вершину только по одному разу и возвращается обратно в исходную вершину.

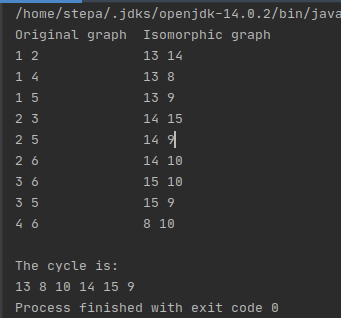
Суть работы протокола в том, что на каждой итерации формируется граф изоморфный данному. Затем в случайном порядке, или по просьбе проверяющей стороны показывается либо Гамильтонов цикл в изоморфном графе, либо изоморфный граф расшифровывается до его изначального состояния(исходного графа). При этом правдивость Алисы доказывается в первом случае существованием цикла, а во втором случае возможностью преобразовать изоморфный граф обратно в исходный. Таким образом, при достижении определённого количества итераций, проверяющая сторона(Боб) убеждается в том, что Алиса владеет Гамильтоновым циклом исходного графа. Вероятность того, что Алиса им не владеет при этом равна 1/(2^k), где k - число итераций.

Пример работы программы:

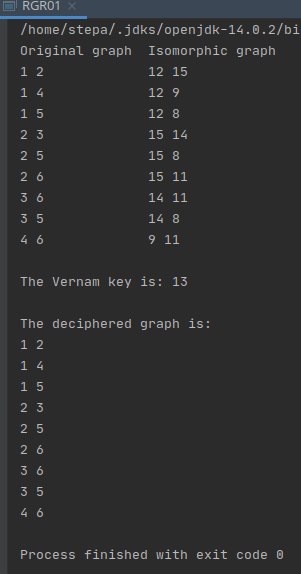
Оригинальный граф и Гамильтонов цикл:



Раскрывается цикл:



Раскрывается(расшифровывается) граф:



Листинг программы:

package com.company;  
  
import java.nio.file.Files;  
import java.nio.file.Path;  
import java.util.ArrayList;  
import java.util.Random;  
  
public class RGR01 {  
 public RGR01(){  
 rnd = new Random();  
 keyVernam = rnd.nextInt(20);//Размер ключа должен быть такого же порядка, что и количество рёбер  
 }  
 ArrayList<Vertice> vertices;//  
 ArrayList<Edge> edges;//Рабочий массив рёбер  
 public int verticesNum;  
 public int edgesNum;  
 public String[] tokens;  
 public int[] hamiltonianCycle;  
 ArrayList<Edge> edgesIsomorphic;  
 public int[] hamiltonianCycleIsomorphic;  
 private int keyVernam;//Ключ для шифрования Вернама  
 Random rnd;  
  
 public static void main(String[] args) {  
 RGR01 rgr = new RGR01();  
 rgr.tokens = rgr.readAFile();//прочитал файл  
 rgr.createVerticesEdgesAndReadHamiltonianCycle();//создал вершины и рёбра в соответствиии с текстом вводимого файла  
 rgr.createIsomorphicGraphAndItsHamiltonianCycle();//создал изоморфный граф и то же, что и выше для исходного  
 // rgr.printAll();  
 rgr.printOriginalAndIsomorphicGraphs();  
 rgr.checkTheGraph();//провёл один цикл проверки с нулевым знанием  
 }  
public void printOriginalAndIsomorphicGraphs(){  
 System.*out*.println("Original graph Isomorphic graph");  
 for (int i = 0; i < edgesNum; i++) {  
 System.*out*.println(edges.get(i).v1Int + " " + edges.get(i).v2Int + " " + edgesIsomorphic.get(i).v1Int + " " + edgesIsomorphic.get(i).v2Int);  
 }  
}  
 public void checkTheGraph(){  
 int ZeroOrOne = rnd.nextInt(2);//Рандомно(произвольно) решаю, показать цикл или изоморфный граф  
 if(ZeroOrOne == 0){  
 System.*out*.println("\nThe cycle is: ");  
 for (int i = 0; i < verticesNum; i++) {//если выпал ноль, показываю цикл  
 System.*out*.print(hamiltonianCycleIsomorphic[i] + " ");  
 }  
 }else{//если выпала единица, показываю ключ и расшифровываю изоморфный граф  
 System.*out*.println("\nThe Vernam key is: " + keyVernam);  
 System.*out*.println("\nThe deciphered graph is: ");  
 for (int i = 0; i < edgesNum; i++) {  
 System.*out*.println(decipherVernam(edgesIsomorphic.get(i).v1Int) + " " + decipherVernam(edgesIsomorphic.get(i).v2Int));  
 }  
 }  
 }  
 public int cipherVernam(int m) {//обычное шифрование Вернама  
 return m ^ keyVernam;  
 }  
 public int decipherVernam(int m){//дешифрование  
 return m ^ keyVernam;  
 }  
  
 public void printAll() {  
 System.*out*.println("Original graph Isomorphic graph");  
 for (int i = 0; i < edgesNum; i++) {  
 System.*out*.println(edges.get(i).v1Int + " " + edges.get(i).v2Int + " " + edgesIsomorphic.get(i).v1Int + " " + edgesIsomorphic.get(i).v2Int);  
 }  
 System.*out*.println("Original Hamiltonian Cycle: ");  
 for (int i = 0; i < verticesNum; i++) {  
 System.*out*.print(hamiltonianCycle[i] + " ");  
 }  
 System.*out*.println("\nCiphered(isomorphic) Hamiltonian cycle: ");  
 for (int i = 0; i < verticesNum; i++) {  
 System.*out*.print(hamiltonianCycleIsomorphic[i] + " ");  
 }  
 System.*out*.println("\nA key for Vernam deciphering is: " + keyVernam);  
 }  
  
 public void createIsomorphicGraphAndItsHamiltonianCycle() {  
 edgesIsomorphic = new ArrayList<>(edgesNum);  
 for (int i = 0; i < edgesNum; i++) {//для создания изоморфного графа просто шифрую номера вершин исходного. В идеале их(рёбра) можно было бы и перетасовать...  
 int v1 = cipherVernam(edges.get(i).v1Int) ;  
 int v2 = cipherVernam(edges.get(i).v2Int) ;  
 edgesIsomorphic.add(i, new Edge(v1, v2));  
 }  
 hamiltonianCycleIsomorphic = new int[edgesNum];  
 for (int i = 0; i < verticesNum; i++) {//также шифрую Гамильтонов цикл  
 hamiltonianCycleIsomorphic[i] = cipherVernam(hamiltonianCycle[i]);  
 }  
 }  
  
 public void createVerticesEdgesAndReadHamiltonianCycle() {//создаю объекты рёбер и вершин в соответствии с текстом входного файла  
 verticesNum = Integer.*parseInt*(tokens[0]);  
 edgesNum = Integer.*parseInt*(tokens[1]);  
 this.vertices = new ArrayList<>(verticesNum);  
 this.edges = new ArrayList<>(edgesNum);  
 for (int i = 0; i < verticesNum; i++) {  
 vertices.add(i, new Vertice());  
 }  
 for (int i = 0; i < edgesNum \* 2; i += 2) {  
 edges.add(new Edge(Integer.*parseInt*(tokens[2 + i]), Integer.*parseInt*(tokens[3 + i])));  
 }  
 for (Edge edge : edges) {  
 // System.out.println(edge.name);  
 }  
 hamiltonianCycle = new int[verticesNum];  
 for (int i = 0; i < verticesNum; i++) {  
 hamiltonianCycle[i] = Integer.*parseInt*(tokens[Integer.*parseInt*(tokens[1]) \* 2 + i + 2]);  
 // System.out.println(" hamiltonianCycle[i] = " + hamiltonianCycle[i]);  
 }  
 }  
  
 public String[] readAFile() {//Читаю файл, засовываю в массив строк его содержимое  
 Path path = Path.*of*("/home/stepa/Documents/projetcs/ZI\_01/src/resources/graph.txt");  
 // System.out.println(path);  
 boolean exists = Files.*exists*(path);  
 //System.out.println("exists = " + exists);  
 String s;  
 String[] tokens = new String[1];  
 try {  
 s = Files.*readString*(path);// UTF 8  
 // System.out.println("s = \n" + s);  
 String delims = "[ \n]+";  
 tokens = s.split(delims);  
  
 } catch (Exception e) {  
 }  
 //System.out.println("tokens = \n");  
 for (String str : tokens  
 ) {  
 //System.out.println(str);  
 }  
 return tokens;  
 }  
  
class Edge {//Рабочий класс)  
 public Edge(int v1, int v2) {  
 this.v1Int = v1;  
 this.v2Int = v2;  
 this.name = v1 + " " + v2;  
 }  
  
 String name;  
 public Vertice v1;  
 public Vertice v2;  
 public int v1Int;  
 public int v2Int;  
}  
  
class Vertice {//Этот класс оказался не нужен, но мог бы пригодиться...  
 String name;  
}