Федеральное агентство связи

Ордена Трудового Красного знамени

Федеральное государственное бюджетное образовательное учреждение

Высшего образования

«Московский технический университет связи и информатики

Кафедра «Математической кибернетики и информационных технологий»

Отчёт по лабораторной работе №1

Организация таблиц идентификаторов

Выполнил:

Студент группы БФИ 1701

Сизов А.Д

Проверила:

Ассистент кафедры МКиИТ

Алексанян Диана Ашотовна

Вариант 13

Москва 2020г.

**Задание**

Написать программу, которая получает на вход набор идентификаторов. Организовать таблицы идентификаторов с помощью методов: простое рехэширование, упорядоченный список. Сравнить эффективность этих методов.

Код класса Main.java

|  |
| --- |
| import java.io.BufferedReader; import java.io.FileReader; import java.io.IOException; import java.util.ArrayList; import java.util.List; import java.util.Scanner;  public class Main {   public static void main(String[] args) throws IOException  {  Scanner in = new Scanner(System.in);  Chains chain = new Chains(300);  String[] array = ReadText();  HashTable HT = new HashTable(300);  int counterRe = 0;  int counterChain = 0;  for (int i = 0; i < array.length; i++)  {  System.out.println(array[i]);  counterRe = counterRe + HT.insertSimpleRehash(array[i]);  counterChain = counterChain + chain.insertChains(array[i]);  }  HT.print();  System.out.println("Колличество коллизий при обычном рехэшировании = " + counterRe);  System.out.println("Колличество коллизий при методе Цепочек = " + counterChain);   //1  System.out.println("МЕТОД ЦЕПОЧЕК");  System.out.println("Введите слово для поиска в методе Цепочек");  String wrd = in.nextLine();  long startTime = System.nanoTime();  boolean CC = chain.checkChain(wrd);  long endTime = System.nanoTime();  long time = endTime - startTime;  System.out.println("Скорость выполнения поиска в методе Цепочек = " + time);  if (CC == true) {System.out.println("Введенное слово есть в таблице");}  else {System.out.println("Введенного слова нет в таблице");}   //2  System.out.println("МЕТОД ПРОСТОГО РЕХЭШИРОВАНИЯ");  System.out.println("Введите слово для поиска в методе Цепочек");  wrd = in.nextLine();  startTime = System.nanoTime();  CC = HT.checkSR(wrd);  endTime = System.nanoTime();  time = endTime - startTime;  System.out.println("Скорость выполнения поиска в методе простого рехэширования = " + time);  if (CC == true) {System.out.println("Введенное слово есть в таблице");}  else {System.out.println("Введенного слова нет в таблице");}  }   //Функция чтения слов из текстового файла  private static String[] ReadText() throws IOException  {  FileReader fileReader = new FileReader("./t.txt");  BufferedReader bufferedReader = new BufferedReader(fileReader);  List<String> lines = new ArrayList<String>();  String line = null;  while ((line = bufferedReader.readLine()) != null) {  lines.add(line);  }  bufferedReader.close();  return lines.toArray(new String[lines.size()]);  }   } |

Код класса Item

|  |
| --- |
| //Класс описывающий элементы, хранящиеся в массиве class Item{   private String key;   public Item(String key)  {  this.key = key;  }   public String getKey() {  return key;  }   public void setKey(String key) {  this.key = key;  } } |

Код класса HashTable

|  |
| --- |
| public class HashTable {   //массив для хранения элементов  private Item[] table;   //размер таблицы  private int size;   //Вызываем конструктор, который создает начальную таблицу нужного размера  public HashTable(int size) {  this.size = size;  table = new Item[size];  }   //Функция получения хэша слова  private int makehashSR(String word)  {  int hash = 0;   for(int i = 0; i < word.length(); i++)  hash = (31 \* hash + word.charAt(i)) % size;   return hash;  }   //Функция заполнения таблицы и простого рехэширования  public int insertSimpleRehash (String key)  {  Item item = new Item(key);  int count = 0;  int hash = makehashSR(key);  while(table[hash] != null)  {  hash = (hash + 31) % size;  count++;  }  table[hash] = item;  return count;  }   //Вывод массива  public void print()  {  for(int i = 0; i < size; i++)  if(table[i] != null)  System.*out*.println(i + " " + table[i].getKey());  }   //Наличие слова в хэш-таблице  public boolean checkSR (String word)  {  int hash = makehashSR(word);  return table[hash].getKey().equals(word)== true;  }  } |

Код класса Chains

|  |
| --- |
| import java.awt.\*; import java.util.ArrayList; import java.util.LinkedList;  public class Chains {  private int[] Pointers; //Массив указателей  private int size;  private ArrayList<ArrayList<String>> Values;//Двумерный список   //Создаем начальные массивы и заполняем массив указателей (-1)  public Chains(int size)  {  this.size = 0;  Pointers = new int[size];   for (int i = 0; i< Pointers.length; i++)  {  Pointers[i] = -1;  }   Values = new ArrayList<>();  }   //Заполнение массивов указателей и двумерного списка слов  public int insertChains (String word)  {  int hash = makeHash(word);  int pointer = Pointers[hash];  int count = 0;  if (pointer == -1)  {  Pointers[hash] = Values.size();  Values.add(new ArrayList<String>());  int vldx = Values.size()-1;  Values.get(vldx).add(word);  this.size+=1;  }  else  {  count ++;  if (Values.get(pointer).contains(word)){ return count;}  else {Values.get(pointer).add(word); return count;}   }  return count;   }   private int makeHash(String str)  {  int hash = 0;   for(int i = 0; i < str.length(); i++)  hash = (31 \* hash + str.charAt(i)) % Pointers.length;   return hash;   }   public boolean checkChain (String word)  {  int hash = makeHash(word);  if (Pointers[hash] == -1) {return false;}  return (Values.get(Pointers[hash]).contains(word));  }       } |

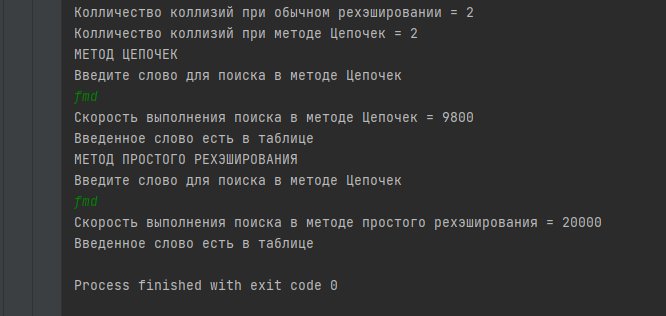


Рисунок 1 – Результат работы программы