МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по учебной практике

Тема: Визуализация алгоритма Ахо-Корасик на языке Java

Студент гр. 7304 _____ Попов С.А.

Студент гр. 7304 _____ Субботин А.С.

Студент гр. 7304 _____ Запевалов А.И.

Руководитель _____ Ефремов М.А.

Санкт-Петербург
2018

ЗАДАНИЕ

НА УЧЕБНУЮ ПРАКТИКУ

Студент Субботин А.С. группы 7304

Студент Запевалов А.И. группы 7304

Тема практики: Визуализация алгоритма Ахо-Корасик на языке Java

Задание на практику:

Командная итеративная разработка визуализатора алгоритма(ов) на Java с графическим интерфейсом.

Алгоритм: Ахо-Корасик

Сроки прохождения практики: 01.07.2019 – 14.07.2019

Дата сдачи отчета: 12.07.2019

Дата защиты отчета: 12.07.2019

Студент гр. 7304	_____	Субботин А.С.
Студент гр. 7304	_____	Попов С.А.
Студент гр. 7304	_____	Запевалов А.И.
Руководитель	_____	Ефремов М.А.

АННОТАЦИЯ

На данную учебную практику поставлено задание визуализации алгоритма Ахо-Корасик на языке Java. В ходе выполнения реализована логика алгоритма и логика отображения автомата Ахо-Корасик.

Для реализации последнего использованы библиотеки SWING и AWT, а также использованы некоторые конструкции многопоточных приложений.

Для тестирования использована библиотека JUnit4.

SUMMARY

For this educational practice, the task of visualization of the algorithm of Aho-Corasick on Java programming language was set. During the development, the logic of algorithm and the logic of visualization of automate of Aho-Corasick were implemented.

To implement the latter, SWING and AWT libraries were used along with some multi-threaded programming constructions.

For the testing purposes JUnit4 library was used.

Содержание

Введение 5
1. Требования к программе 6
2. План разработки и распределение ролей в бригаде 7
   2.1. План разработки 7
   2.2. Распределение ролей в бригаде 7
3. Особенности реализации 8
   3.1. Теоретические сведения 8
   3.2. Построение бора 8
   3.3. Переходы по бору и суффиксные ссылки 9
   3.4. Сжатые суффиксные ссылки 11
   3.5. Использование автомата 11
   3.6. Основные методы 13
4. Тестирование 14
   4.1 Тестирование алгоритма 14
Заключение 15
Список использованных источников 16
ПРИЛОЖЕНИЕ А 17

ВВЕДЕНИЕ

Цель работы – изучить основы программирования на Java:

- Изучить базовые конструкции языка
- Изучить реализацию парадигмы ООП на Java
- Изучить использование библиотеки SWING
- Изучить создание юнит-тестов на Java

Задача – визуализировать работу алгоритма Ахо-Корасик на языке Java.

В ходе выполнения работы были изучены основы языка программирования Java, использование библиотек Swing, AWT, исследованы возможность встроенных классов коллекций, изучены некоторые проблемы синхронизации многопотоковых приложений.

В результате создано приложение с графическим интерфейсом, способное выполнять логику алгоритма Ахо-Корасик и отображать состояние автомата на каждом шаге выполнения.

1. ТРЕБОВАНИЯ К ПРОГРАММЕ

1.1. Исходные требования к программе

Программа должна быть способна выполнять следующие действия:

- Построение автомата Ахо-Корасик для данного множества шаблонов
- Представление автомата Ахо-Корасик в виде графа
- Пошаговое выполнение алгоритма Ахо-Корасик для данной строки и данного множества подстрок
- Визуализация состояния автомата на каждом шаге

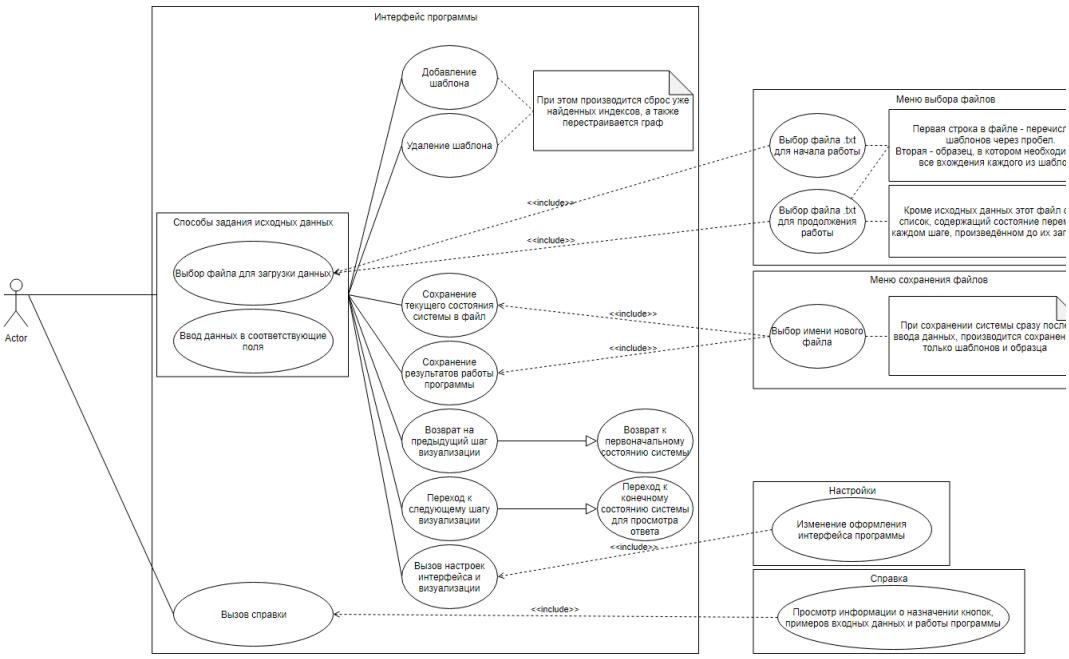

Рисунок 1 - Диаграмма Use-case

1.2. Уточнение требований после сдачи прототипа

Связать вместе логику алгоритма и графический интерфейс, обеспечить интерактивность отображению автомата.

1.3. Уточнение требований сдачи 1-й версии

Добавить диаграмму классов, файл team.md и Unit-тесты. Реализовать pom.xml и привести структуру проекта к общему виду.

1.4. Уточнение требований сдачи 2-й версии

Исправить ошибки логики работы программы, написать отчет.

2. ПЛАН РАЗРАБОТКИ И РАСПРЕДЕЛЕНИЕ РОЛЕЙ В БРИГАДЕ

2.1. План разработки

01/07/18 – поставить задание, распределить роли. Создать примерный план разработки. Создать первую версию диаграммы Use case.

02/07/18 – создание классы Bohr и Node, описание методов для перехода по автомату. Реализовать ввод и вывод с консоли.

03/07/18 – довести класс Bohr до работоспособного состояния.

04/07/18 – создать класс AlgorithmTest, забирающий себе управление автоматом и выполняющий действия алгоритма Ахо-Корасик.

05/07/18 – произвести дальнейшую отладку класса AlgorithmTest. Реализовать возможность ввода из текстового поля в окне.

06/07/18 – создать прототип интерфейса окна вывода(класс AnswerFrame) и окна визуализации(классы VisualiseUI, Graph, CoordinateGenerator и Point).

07/07/18 – улучшить интерфейс визуализации. Добавить возможность пошагового перехода по алгоритму вперед и назад.

08/07/18 – исправить ошибки ввода данных и визуализации. Написать тесты класса AlgorithmTest.

09/07/18 – Первая версия отчета, создание диаграммы классов и исправление интерфейса.

12/07/18 – Исправленная версия отчета, диаграммы классов и классов тестирования.

2.2. Распределение ролей в бригаде

- Запевалов Алексей – алгоритм, UI, отчёт.
- Субботин Антон — визуализация графа, UI, отладка, Unit-тестирование.
- Попов Сергей — визуализация графа.

3. ОСОБЕННОСТИ РЕАЛИЗАЦИИ

3.1. Теоретические сведения

Есть множество шаблонов $\Phi = \{p_1, \dots, , p_n\}$ и один текст $|T| = n$.

Суммарная длина всех образцов - m.

Нужно найти, какие образцы из данного множества встречаются в этом тексте. Применение алгоритма Кнута-Морриса-Пратта неэффективно, т.к. алгоритм нужно будет запустить для каждого текста.

3.2. Построение бора

Бор - дерево с корнем в некоторой вершине, каждое ребро которого подписано некоторой буквой.

Пусть дан набор строк - {he, she, his, hers}

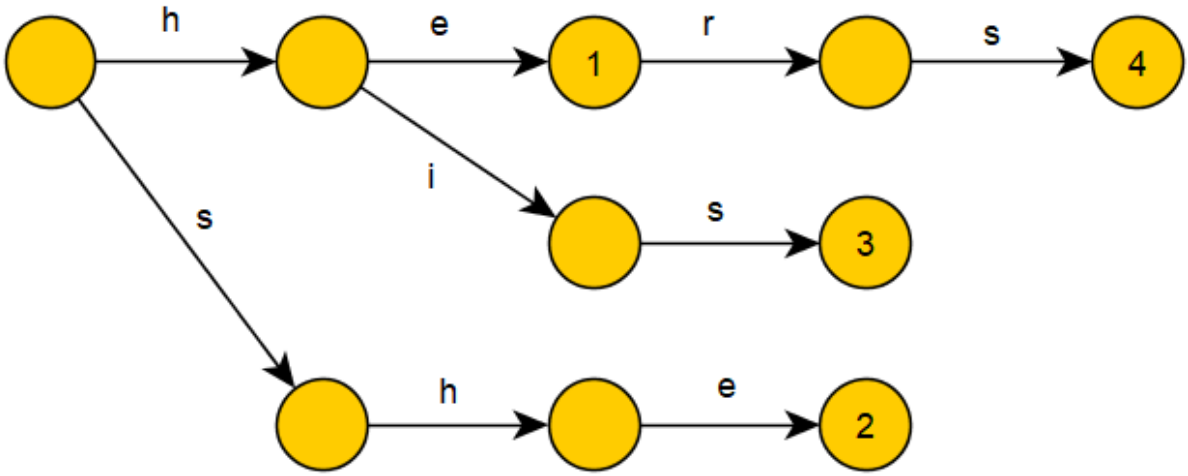

Рисунок 2 – Пример бора

У дерева каждая дуга имеет пометку в виде одного символа, и не может быть двух дуг из одной вершины, которые имеют одинаковую пометку.

Элемент бора на псевдокоде:

```
struct Node:
    Node son[k]                     // массив сыновей
    Node go[k]                      // массив переходов (запоминаем
переходы в ленивой рекурсии), используемый для вычисления суффиксных
ссылок
    Node parent                     // вершина родитель
```

```
    Node suffLink                    // суффиксная ссылка (вычисляем в
ленивой рекурсии)
    Node up                          // сжатая суффиксная ссылка
    char charToParent                // символ, ведущий к родителю
    bool isLeaf                      // флаг, является ли вершина
терминалом
    vector<int> leafPatternNumber    // номера строк, за которые
отвечает терминал
```

Добавление слова в бор на псевдокоде:

```
fun addString(string s, int patternNumber):
    Node cur = root
    for i = 0 to s.length - 1
        char c = s[i]
        if cur.son[c] == 0
            cur.son[c] = Node
            /* здесь также нужно обнулить указатели на переходы
и сыновей */
            cur.son[c].suffLink = 0
            cur.son[c].up = 0
            cur.son[c].parent = cur
            cur.son[c].charToParent = c
            cur.son[c].isLeaf = false
        cur = cur.son[c]
    cur.isLeaf = true
    cur.leafPatternNumber.pushBack(patternNumber)
```

Пусть $[u]$ – слово, приводящие в вершину u в боре. Узлы бора можно понимать как состояния автомата, корень - как начальное состояние. Соответственно, узлы бора, в которых заканчиваются строки, становятся терминальными.

3.3. Переходы по бору и суффиксные ссылки

Заведем несколько функций для перехода по бору:

- $parent(u)$ – возвращает родителя вершины u
- $\pi(u) = \delta(\pi(parent(u)), c)$ – суффиксная ссылка, и существует переход из $parent(u)$ в u по символу c

Суффиксная ссылка для вершины u – вершина, в которой оканчивается наидлиннейший собственный суффикс строки, соответствующей вершине u. Суффикс может быть и нулевой длины - т.е. суффиксная ссылка может вести в корень.

Другими словами, $\pi(u) = v$, если $[v]$ – максимальный суффикс $[u]$, $[u] \neq [v]$.

Суффиксные ссылки для вышеуказанного примера:

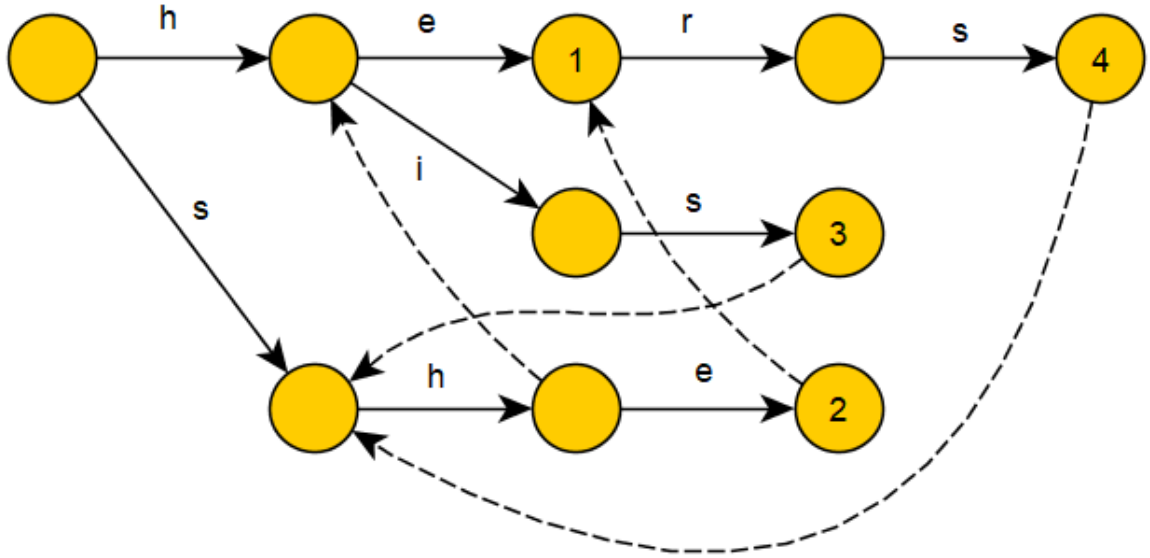

Рисунок 3 – Пример бора с суффиксными ссылками

Если ссылка не обозначена, значит, она ведёт в корень.

Переход по бору:

$$\delta(u,c) = \begin{cases} v & \text{если } v \text{ – сын символа } c \\ root & \text{если } u \text{ – корень и } c \text{ – не сын } u \\ \delta(\pi(u),c) & \text{иначе} \end{cases}$$

Переход осуществляется по текущей вершине u и символу c.

Функция перехода на псевдокоде

```
Node getLink(Node v, char c):
   if v.go[c] == null  // если переход по символу c ещё не вычислен
        if v.son[c]
            v.go[c] = v.son[c]
        else if v == root
            v.go[c] = root
        else
            v.go[c] = getLink(getSuffLink(v), c)
    return v.go[c]
```

3.4. Сжатые суффиксные ссылки

При построении автомата может возникнуть ситуация, что ветвление есть не на каждом символе. Тогда можно использовать **сжатые суффиксные ссылки:**

$$up(u) = \begin{cases} \pi(u) & \text{если } u - \text{терминальный} \\ \emptyset & \text{если } u - \text{корень} \\ up(\pi(u)) & \text{иначе} \end{cases}$$

$\mathrm{up(u)} -$ ближайшее допускающее состояние (терминал) перехода по суффиксным ссылкам.

Вычисление сжатых суффиксных ссылок на псевдокоде:

```
Node getUp(Node v):
    if v.up == null // если сжатая суффиксная ссылка ещё не вычислена
        if getSuffLink(v).isLeaf
            v.up = getSuffLink(v)
        else if getSuffLink(v) == root
            v.up = root
        else
            v.up = getUp(getSuffLink(v))
    return v.up
```

В результате вышеописанных действий построен конечный детерминированный автомат.

3.5. Использование автомата

В общих чертах, получившийся автомат нужно использовать следующим образом: по очереди просматривать символы текста, для каждого символа 'c' осуществляя переход по $\delta(\mathrm{u,c}),$ где $\delta -$ текущее состояние. Оказавшись в новом состоянии, отметить по сжатым суффиксным ссылкам строки, которые встретились, и, если требуется, позицию.

Использование автомата на псевдокоде:

```
fun processText(string t):
    Node cur = root
    for i = 0 to t.length - 1
        char c = t[i] - 'a'
```

```
        cur = getLink(cur, c)
        /* В этом месте кода должен выполняться переход по сжатой
суффиксной ссылке getUp(cur). Для вершины,
           обнаруженной по ней тоже ставим, что она найдена, затем
повторяем для её сжатой суффиксной ссылки
           и так до корня. */
```

3.6. Основные методы

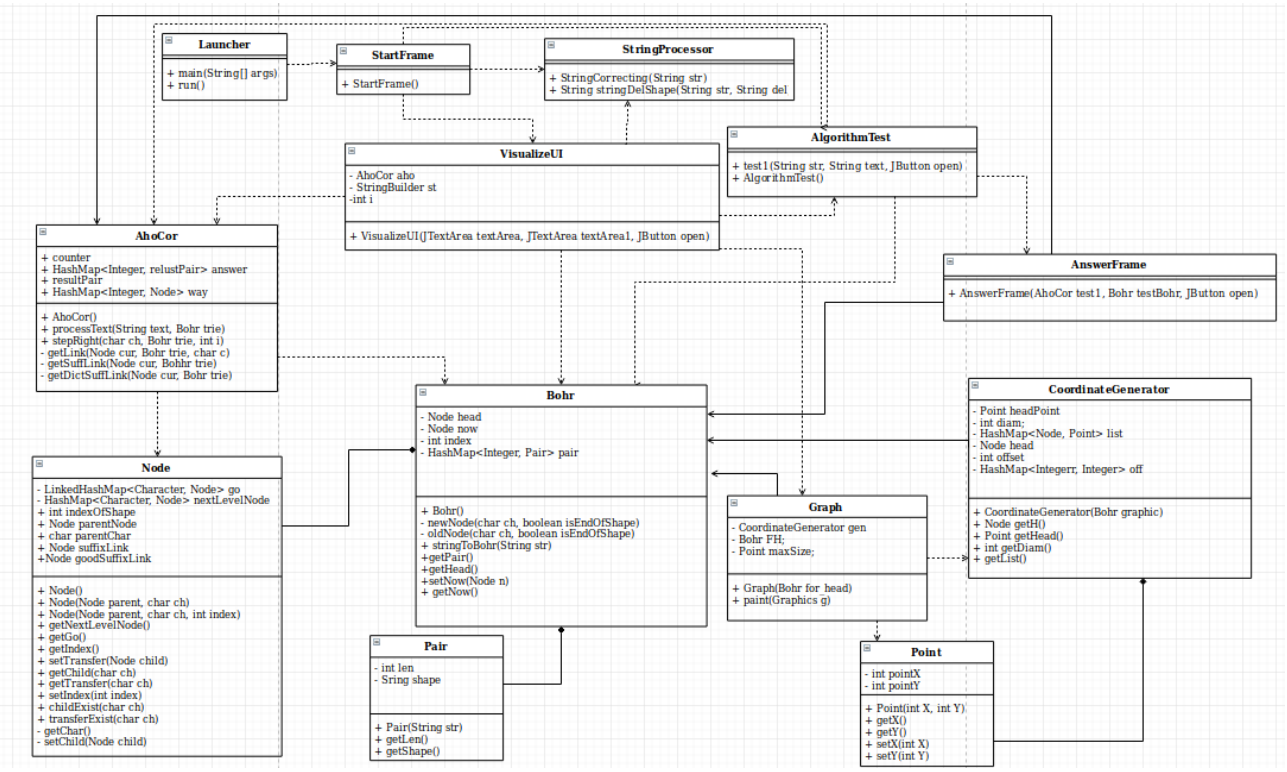

Рисунок 4 - UML-диаграмма проекта

Рисунок 5 – Интерфейс программы

4. ТЕСТИРОВАНИЕ

4.1 Тестирование алгоритма

Для тестирования алгоритма подключена библиотека JUnit4. Написан ряд юнит-тестов, проверяющих корректную работу каждого класса алгоритма. Их корректная работа свидетельствует об отсутствии явных ошибок в реализации.

Класс AhoCorTest используется для проверки алгоритма ахо-корасик. В методе processText() проверяется выводимый текст — его размер и содержание. В методе stepRight() проверяется нахождение потомков в боре.

Класс BohrTest служит для проверки правильности построения бора. В методе stringToBohr() добавляется несолько узлов на разных уровнях, после чего проверяется их связь с корнем бора. В методе getPair() проверяется соответствие терминальных узлов ожидаемым. В методе getHead() находится проверка размера одного из уровней бора. В методе getNow() проверяется правильность вызова указателя на текущий узел.

Класс NodeTest() используется для проверки нахождения различных узлов бора методами программы. В методе getNextLevelNode() происходит проверка правильности добавления и нахождения новых элементов в боре. В методе getIndex() - проверка порядкового номера данного шаблона. В методе transferExist() происходит проверка наличия перехода из массива переходов узла. В методе getChild() - проверка наличия и отсутствия заданного потомка у заданного узла. В методе getGo() - проверка узлов по заданным переходам.

При работе была использована система контроля версий Git вместе с сервисом GitHub.

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной работы изучены базовые конструкции языка Java и реализация парадигмы ООП на этом языке. Исследованы возможности библиотеки SWING для создания графического интерфейса и Junit4 для создания юнит-тестов на Java.

Изучены некоторые возможности языка Java, такие как: встроенные классы коллекций, работа с файловой системой, многопоточность, события.

Как результат, построено приложение с графическим интерфейсом, способное выполнять алгоритм Ахо-Корасик и отображать состояние алгоритма на каждом шаге выполнения.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Шилдт Г. SWING. Руководство для начинающих. М.: Издательство Вильямс, 2007. 705 с.
2. Липский В. Комбинаторика для программистов. М.: Мир, 1988, 200 с.
3. Алгоритм Ахо-Корасик // Викиконспекты ИТМО. URL: https://neerc.ifmo.ru/wiki/index.php?title=%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%90%D1%85%D0%BE-%D0%9A%D0%BE%D1%80%D0%B0%D1%81%D0%B8%D0%BA.
4. Кей Хорстманн, "Java. Библиотека профессионала", Том II, Тонкости программирования. М.: Издательство Вильямс, 2007. 1168 с.
5. Лекции Корытова Павла, 6304 // OneNote Online. URL: https://onedrive.live.com/edit.aspx?cid=37598f7bb5ca9f32&page=view&resid=37598F7BB5CA9F32!1239&parId=37598F7BB5CA9F32!106&authkey=!AN4Skxub7IIh0Wc&app=OneNote.

ПРИЛОЖЕНИЕ А

```
//Bohr.java
package io.desmy.project;
import java.util.HashMap;
public class Bohr {
	private Node head;
	private Node now;
	private int index;
	private HashMap<Integer, Pair> pair;
	public class Pair{
		private int len;
		private String shape;
		public Pair(String str){
			shape = str;
			len = shape.length();
		}
		public int getLen() { return len; }
		public String getShape() { return shape; }
	}
	public Bohr() {
		head = new Node();
		pair = new HashMap<>();
		now = head;
		index = 0;
	}
	private void newNode(char ch, boolean isEndOfShape) {
		if(isEndOfShape) {
			new Node(now, ch, index);
			now = head;
		}
		else
			now = new Node(now, ch);
	}
	private void oldNode(char ch, boolean isEndOfShape) {
		now = now.getChild(ch);
		if(isEndOfShape) {
			now.setIndex(index);
			now = head;
		}
	}
	public void stringToBohr(String str) {
		str += ' ';
		int i = 1;
		char ch;
		while(i < str.length()) {
			ch = str.charAt(i-1);
			if (now.childExist(ch))
				oldNode(ch, str.charAt(i) == ' ');
			else
				newNode(ch, str.charAt(i) == ' ');
			if(str.charAt(i) == ' ')
				index++;
			while(i < str.length() && str.charAt(i) == ' ')
				i++;
			i++;
		}
	}
	public void stringToPair(String str) {
		StringBuilder text = new StringBuilder(str);
```

```
            text.append(' ');
            int j, k = 0;
            for(int i = 0; i < text.length(); i++)
                if(text.charAt(i) != ' ') {
                    for (j = i; text.charAt(j) != ' '; j++);
                    pair.put(k++, new Pair(text.substring(i, j)));
                    i = j;
                }
        }
        public HashMap<Integer, Pair> getPair() {
            return pair;
        }
        public Node getHead() { return head; }
        public void setNow(Node n) { now = n; }
        public Node getNow() { return now; }
}
```

//ahoCor.java

```
package io.desmy.project;
import java.util.HashMap;
/**
 * функция processText проходит по тексту, использует по одному символу из него.
 * каждый раз происходит вызов поиска перехода и поиска сжатой суффиксной ссылки.
 */
public class AhoCor {
	public int counter = 0;
	public HashMap<Integer, resultPair> answer;
	public AhoCor(){
		answer = new HashMap<>();
		way = new HashMap<>();
	}
	public class resultPair {
		int position;
		int template;
		public resultPair(int pos, int temp) {
			position = pos;
			template = temp;
		}
		public int getPosition() { return position; }
		public int getTemplate() { return template; }
	}
	public void processText(String text, Bohr trie) {
		Node cur = trie.getHead();
		Node dsl;
		int resultCounter = 0;
		for (int i = 0; i < text.length(); i++){
			char c = text.charAt(i);
			cur = getLink(cur, trie, c); //поиск перехода
			dsl = cur;
			do {
				if (dsl.getIndex() != -1) {
					answer.put(resultCounter, new resultPair( i + 1, dsl.getIndex()));
					resultCounter++;
				}
				dsl = getDictSuffLink(dsl, trie); //поиск сжатой суффиксной ссылки
			} while (dsl != trie.getHead());
		}
	}
	public HashMap<Integer, Node> way;
	public void stepRight(char ch, Bohr trie, int i) {
		way.put(i, trie.getNow());
		Node cur = getLink(trie.getNow(), trie, ch);
		if(cur == trie.getHead())
			return;
		trie.setNow(cur);
		Node dsl = cur;
		do {
			if(dsl.getIndex() != -1)
				answer.put(counter++, new resultPair(i+1, dsl.getIndex()));
			dsl = getDictSuffLink(dsl, trie); //поиск сжатой суффиксной ссылки
		} while (dsl != trie.getHead());
	}
	/**
	 * Переход к ребёнку
	 * * Если есть - возвращаем на него указатель
```

```
     *  * Если нет:
     *  * 1 Если родитель - корень, возвращаем корень
     *  * 2 Если нет - находим суффикс поменьше
     * @param cur - текущий узел
     * @param trie - бор
     * @param c - символ ребенка
     * @return Node
     */
    private Node getLink(Node cur, Bohr trie, char c) {
        if (!cur.transferExist(c)){
            if (cur.childExist(c)){
                cur.setTransfer(cur.getChild(c));
            }
            else if (cur == trie.getHead()) {
                cur.setTransfer(trie.getHead());
            }
            else {
                cur.setTransfer(getLink(getSuffLink(cur, trie),trie, c));
            }
        }
        if (cur.transferExist(c)){
            return cur.getTransfer(c);
        }
        else{
            return trie.getHead();
        }
    }
    /**  Поиск суффиксной ссылки
     * Пока не найден ребенок по символу
     * исследуем суффиксы пока не дошли до корня
     */
    private Node getSuffLink(Node cur, Bohr trie) {
        if (cur.suffixLink == null){
            if (cur == trie.getHead() || cur.parentNode == trie.getHead()){
                cur.suffixLink = trie.getHead();
            }
            else{
                cur.suffixLink = getLink(getSuffLink(cur.parentNode, trie), trie,
cur.parentChar);
            }
        }
        return cur.suffixLink;
    }
    /* Поиск сжатой суффиксной ссылки
     * пока очередная суффиксная ссылка не привела в терминал или корень
     */
    private Node getDictSuffLink(Node cur, Bohr trie) {
        if (cur.goodSuffixLink == null)
        {
            if (getSuffLink(cur, trie).indexOfShape != -1){
                cur.goodSuffixLink = getSuffLink(cur, trie);
            }
            else if (getSuffLink(cur, trie) == trie.getHead()){
                cur.goodSuffixLink = trie.getHead();
            }
            else{
                cur.goodSuffixLink = getDictSuffLink(getSuffLink(cur, trie), trie);
            }
        }
```

```
            return cur.goodSuffixLink;
        }
    }
```

```
//Node.java

package io.desmy.project;

import java.util.HashMap;
import java.util.LinkedHashMap;
public class Node {
	private LinkedHashMap<Character, Node> go = new LinkedHashMap<>();
	private HashMap<Character, Node> nextLevelNode = new HashMap<>();  //Создание
с нуля
	public int indexOfShape;  //Вход, если последний символ
	public Node parentNode;  //Вход
	public char parentChar;  //Вход (получаем по parentNode)
	public Node suffixLink;  //Пустое значение
	public Node goodSuffixLink;  //Пустое значение
	public Node() { indexOfShape = -1; parentChar = ' '; }
	public Node(Node parent, char ch) {
		this();
		parentNode = parent;
		parentChar = ch;
		parent.setChild(this);
	}
	public Node(Node parent, char ch, int index) {
		this(parent, ch);
		indexOfShape = index;
	}
	public HashMap<Character, Node> getNextLevelNode() { return nextLevelNode; }
	public LinkedHashMap<Character, Node> getGo() { go.remove(' '); return go; }
	public int getIndex() { return indexOfShape; }
	public void setTransfer(Node child) {  go.put(child.getChar(), child); }
	public Node getChild(char ch) { return nextLevelNode.get(ch);  }
	public Node getTransfer(char ch) { return go.get(ch); }
	public void setIndex(int index) { indexOfShape = index; }
	public boolean childExist(char ch) { return nextLevelNode.get(ch) != null; }
	public boolean transferExist(char ch) { return go.get(ch) != null; }
	private char getChar() { return parentChar; }
	private void setChild(Node child) { nextLevelNode.put(child.getChar(), child); }
}
```

```
//StringProcessor.java

package io.desmy.project;

import java.util.HashMap;
public class StringProcessor {
	static public String stringCorrecting(String str) {
		HashMap<String, Integer> pr = new HashMap<>();
		StringBuilder text = new StringBuilder(str.replaceAll("[^A-Za-zA-Яа-я0-9]", " "));
		text.append(' ');
		int j, k = 0;
		for(int i = 0; i < text.length(); i++)
			if(text.charAt(i) != ' ') {
				for (j = i; text.charAt(j) != ' '; j++);
				pr.put(text.substring(i, j), k++);
				i = j;
			}
		StringBuilder txt = new StringBuilder("");
		pr.forEach((key, value) -> {
			txt.append(key);
			txt.append(" ");
		});
		return txt.toString();
	}
	static public String stringDelShape(String str, String del) {
		HashMap<String, Integer> pr = new HashMap<>();
		StringBuilder text = new StringBuilder(str.replaceAll("[^A-Za-zA-Яа-я0-9]", " "));
		text.append(' ');
		int j, k = 0;
		for(int i = 0; i < text.length(); i++)
			if(text.charAt(i) != ' ') {
				for (j = i; text.charAt(j) != ' '; j++);
				pr.put(text.substring(i, j), k++);
				i = j;
			}
		HashMap<String, Integer> prDel = new HashMap<>();
		StringBuilder textDel = new StringBuilder(del.replaceAll("[^A-Za-zA-Яа-я0-9]", "
"));
		textDel.append(' ');
		k = 0;
		for(int i = 0; i < textDel.length(); i++)
			if (textDel.charAt(i) != ' ') {
				for (j = i; textDel.charAt(j) != ' '; j++) ;
				prDel.put(textDel.substring(i, j), k++);
				i = j;
			}
		StringBuilder txt = new StringBuilder("");
		prDel.forEach((keyDel, valueDel) -> {
			pr.remove(keyDel);
		});
		pr.forEach((key, value) -> {
			txt.append(key);
			txt.append(" ");
		});
		return txt.toString();
	}
}
```

```
//AlgorithmTest.java

package io.desmy.ui;

import io.desmy.project.*;
import javax.swing.*;
public class AlgorithmTest {
	public AlgorithmTest() {}
	public void test1(String str, String text, JButton open) {
		Bohr testBohr = new Bohr();
		testBohr.stringToBohr(str);
		testBohr.stringToPair(str);
		AhoCor test1 = new AhoCor();
		test1.processText(text, testBohr);
		new AnswerFrame(test1, testBohr, open);
	}
}
```

```
//AnswerFrame.java

package io.desmy.ui;

import io.desmy.project.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
public class AnswerFrame extends JFrame {
	public AnswerFrame(AhoCor test1, Bohr testBohr, JButton open) {
		super("Результат");
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		JPanel panel = new JPanel();
		panel.setLayout(null);
		JTextArea textArea = new JTextArea(10, 1);
		textArea.setEditable(false);
		textArea.setCaretPosition(0);
		textArea.setLineWrap(true);
		textArea.setWrapStyleWord(true);
		JScrollPane scrollPane = new JScrollPane(textArea);

scrollPane.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);
		scrollPane.setSize(new Dimension(400,200));
		panel.add(scrollPane);
		test1.answer.forEach((key, value) -> {
			textArea.append("[" + (value.getPosition() -
testBohr.getPair().get(value.getTemplate()).getLen() + 1)  + " - " +
testBohr.getPair().get(value.getTemplate()).getShape() + "]\n");
		});
		JButton prev = new JButton("Назад");
		prev.setBounds(100,205, 200, 20);
		prev.addActionListener(new ActionListener() {
			@Override
			public void actionPerformed(ActionEvent e) {
				open.doClick();
				dispose();
			}
		});
		panel.add(prev);
		String OS = System.getProperty("os.name").toLowerCase();
		setBounds(300, 300, 400+15*(OS.contains("win") ? 1:0), 269);
		setResizable(false);
		add(panel);
		setLocationRelativeTo(null);
		setVisible(true);
	}
}
```

```
//Launcher.java

package io.desmy.ui;

public class Launcher {
	public static void main(String[] args) {
		javax.swing.SwingUtilities.invokeLater(new Runnable() {
			public void run() {
				try {
					new StartFrame();
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}
}
```

```
//StartFrame.java

package io.desmy.ui;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;
import java.util.Scanner;
import java.io.FileNotFoundException;
import javax.swing.*;
    import io.desmy.project.StringProcessor;
public class StartFrame extends JFrame {
    public StartFrame() {
        super("Выбор входных данных");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JPanel panel = new JPanel();
        panel.setLayout(null);
        JTextArea textArea = new JTextArea(10, 1);
        JTextArea textArea1 = new JTextArea(10, 1);
        JScrollPane scrollShape = new JScrollPane(textArea);
        JScrollPane scrollText = new JScrollPane(textArea1);
        JLabel shapeLabel = new JLabel("Выбранный файл шаблонов");
        JLabel textLabel = new JLabel("Выбранный файл с текстом");
        JButton shapeButton = new JButton("Выбрать файл шаблонов");
        JButton textButton = new JButton("Выбрать текст");
        JButton resultButton = new JButton("Найти вхождения");
        JButton visualButton = new JButton("Визуализация");
        JButton b = new JButton("");
        shapeButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                JFileChooser fileOpen = new JFileChooser();
                int ret = fileOpen.showDialog(null, "Открыть файл");
                if (ret == JFileChooser.APPROVE_OPTION) {
                    File file = fileOpen.getSelectedFile();
                    try (Scanner sc = new Scanner(file,"Cp1251"))  {
                        StringBuilder stringBuilder = new StringBuilder();
                        while (sc.hasNextLine()) {
                            stringBuilder.append(sc.nextLine());
                            stringBuilder.append(" ");
                        }
                        textArea.setText(stringBuilder.toString());
                    } catch (FileNotFoundException en) {
                        System.err.println("File not found. Please scan in new file.");
                    }
                    shapeLabel.setText(file.getName());
                }
            }
        });
        textButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                JFileChooser fileOpen = new JFileChooser();
                int ret = fileOpen.showDialog(null, "Открыть файл");
                if (ret == JFileChooser.APPROVE_OPTION) {
                    File file = fileOpen.getSelectedFile();
                    try (Scanner sc = new Scanner(file, "Cp1251"))  {
                        StringBuilder stringBuilder = new StringBuilder();
                        while (sc.hasNextLine()) {
                            stringBuilder.append(sc.nextLine());
                            stringBuilder.append(" ");
```

```
                        }
                        textArea1.setText(stringBuilder.toString());
                    } catch (FileNotFoundException en) {
                        System.err.println("File not found. Please scan in new file.");
                    }
                    textLabel.setText(file.getName());
                }
            }
        });
        b.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent actionEvent) {
                setVisible(true);
            }
        });
        visualButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent actionEvent) {
                VisualizeUI VI = new VisualizeUI(textArea, textArea1, b);
                setVisible(false);
            }
        });
        resultButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent actionEvent) {
                StringBuilder str = new StringBuilder(textArea.getText());
                while(str.length() != 0 && str.charAt(0) == ' ')
                    str.deleteCharAt(0);
                textArea.setText(str.toString());
                AlgorithmTest newTest = new AlgorithmTest();
                textArea.setText(StringProcessor.stringCorrecting(textArea.getText()));
                newTest.test1(textArea.getText(), textArea1.getText(), b);
                setVisible(false);
            }
        });

scrollShape.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);

scrollText.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);
        textArea1.setCaretPosition(0);
        textArea.setCaretPosition(0);
        textArea1.setLineWrap(true);
        textArea.setLineWrap(true);
        textArea1.setWrapStyleWord(true);
        textArea.setWrapStyleWord(true);
        scrollShape.setBounds(0,0,400, 100);
        shapeButton.setBounds(5,105, 190, 25);
        shapeLabel.setBounds(205,110,190, 15);
        scrollText.setBounds(0,135,400,100);
        textButton.setBounds(5,240,190,25);
        textLabel.setBounds(205,245,190, 15);
        resultButton.setBounds(0,280,200,30);
        visualButton.setBounds(200, 280, 200, 30);
        panel.add(shapeLabel);
        panel.add(textLabel);
        panel.add(scrollText);
        panel.add(scrollShape);
        panel.add(shapeButton);
        panel.add(textButton);
```

```
            panel.add(resultButton);
            panel.add(visualButton);
            add(panel);
            String OS = System.getProperty("os.name").toLowerCase();
            setBounds(300, 300, 400+16*(OS.contains("win") ? 1:0), 349);
            setLocationRelativeTo(null);
            setResizable(false);
            setVisible(true);
        }
    }
```

VisualizeUI.java

```
package io.desmy.ui;

import io.desmy.graph.Graph;
import io.desmy.project.AhoCor;
import io.desmy.project.*;
import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
public class VisualizeUI extends JFrame {
	private AhoCor aho = null;
	private StringBuilder st = null;
	private int i = 0;
	public VisualizeUI(JTextArea textArea, JTextArea textArea1, JButton open) {
		super("Визуализация алгоритма Ахо-Корасик");
		String OS = System.getProperty("os.name").toLowerCase();
		setBounds(300, 300, 600+16*(OS.contains("win") ? 1:0), 369);
		setResizable(false);
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		Bohr testBohr = new Bohr();
		textArea.setText(StringProcessor.stringCorrecting(textArea.getText()));
		testBohr.stringToBohr(textArea.getText());
		testBohr.stringToPair(textArea.getText());
		Graph graph = new Graph(testBohr);
		JScrollPane scrollGraph = new JScrollPane(graph);
		scrollGraph.setSize(300,300);
		add(scrollGraph);
		JPanel panel = new JPanel();
		panel.setLayout(null);
		JButton prevSt = new JButton("Шаг назад");
		JButton nextSt = new JButton("Шаг вперёд");
		JButton inTheEnd = new JButton("Результат");
		JButton addShape = new JButton("Вставить");
		JButton delShape = new JButton("Удалить");
		JButton forOpen = new JButton("К началу");
		JTextField forShape = new JTextField();
		JTextArea forText = new JTextArea(textArea1.getText());
		JTextArea forResult = new JTextArea();
		JScrollPane scrollText = new JScrollPane(forText);
		JScrollPane scrollResult = new JScrollPane(forResult);
		forOpen.setBounds(300, 300, 150, 30);
		prevSt.setBounds(0, 300, 150, 30);
		nextSt.setBounds(150, 300, 150, 30);
		inTheEnd.setBounds(450, 300, 150, 30);
		forShape.setBounds(300, 0, 300, 20);
		addShape.setBounds(300, 20, 150, 20);
		delShape.setBounds(450, 20, 150, 20);
		scrollText.setBounds(300, 40, 300, 130);
		scrollResult.setBounds(300, 170, 300, 130);
		forText.setEditable(false);
		forResult.setEditable(false);
		forText.setLineWrap(true);
		forText.setWrapStyleWord(true);
		forResult.setLineWrap(true);
		forResult.setWrapStyleWord(true);

scrollText.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);
```

```
scrollResult.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);
        JButton back = new JButton("");
        back.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                setVisible(true);
            }
        });
        prevSt.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent actionEvent) {
                if (aho != null && i > 0){
                    testBohr.setNow(aho.way.get(--i));
                    if(i==0)
                        forText.setText(textArea1.getText());
                    aho.way.remove(i);
                    if(i > 0 && i < textArea1.getText().length()) {
                        StringBuilder tex = new StringBuilder(textArea1.getText());
                        StringBuilder buf = new StringBuilder("");
                        buf.append(tex.substring(0,i-1));
                        buf.append("[");
                        buf.append(tex.charAt(i-1));
                        buf.append("]");
                        buf.append(tex.substring(i, tex.length()));
                        forText.setText(buf.toString());
                        forText.setCaretPosition(0);
                    }
                    while(aho.counter > 0 && aho.answer.get(aho.counter -
1).getPosition() == i+1) {
                        aho.answer.remove(--aho.counter);
                    }
                        StringBuilder no = new StringBuilder("");
                        aho.answer.forEach((key, value) -> {
                            no.append("[");
                            no.append(value.getPosition() -
testBohr.getPair().get(value.getTemplate()).getLen() + 1);
                            no.append(" - ");

no.append(testBohr.getPair().get(value.getTemplate()).getShape());
                            no.append("]\n");
                        });
                        forResult.setText(no.toString());
                    graph.rePaint();
                }
                else {
                    forText.setText(textArea1.getText());
                    forText.setCaretPosition(0);
                }
            }
        });
        nextSt.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent actionEvent) {
                if (aho == null) {
                    aho = new AhoCor();
                    st = new StringBuilder(textArea1.getText());
                }
                if (i < st.length()) {
```

```
                        aho.stepRight(st.charAt(i), testBohr, i++);
                        graph.rePaint();
                        if (i == st.length()) {
                            StringBuilder tex = new StringBuilder(textArea1.getText());
                            StringBuilder buf = new StringBuilder("");
                            buf.append(tex.substring(0,i-1));
                            buf.append("[");
                            buf.append(tex.charAt(i-1));
                            buf.append("]");
                            forText.setText(buf.toString());
                            forText.setCaretPosition(0);
                            inTheEnd.doClick();
                        }
                    } else
                        inTheEnd.doClick();
                    StringBuilder no = new StringBuilder("");
                    aho.answer.forEach((key, value) -> {
                        no.append("[");
                        no.append(value.getPosition() -
testBohr.getPair().get(value.getTemplate()).getLen() + 1);
                        no.append(" - ");
                        no.append(testBohr.getPair().get(value.getTemplate()).getShape());
                        no.append("]\n");
                    });
                    forResult.setText(no.toString());
                    if(i > 0 && i < textArea1.getText().length()) {
                        StringBuilder tex = new StringBuilder(textArea1.getText());
                        StringBuilder buf = new StringBuilder("");
                        buf.append(tex.substring(0,i-1));
                        buf.append("[");
                        buf.append(tex.charAt(i-1));
                        buf.append("]");
                        buf.append(tex.substring(i, tex.length()));
                        forText.setText(buf.toString());
                        forText.setCaretPosition(0);
                    }
                }
            });
            inTheEnd.addActionListener(new ActionListener() {
                @Override
                public void actionPerformed(ActionEvent actionEvent) {
                    AlgorithmTest newTest = new AlgorithmTest();
                    textArea.setText(StringProcessor.stringCorrecting(textArea.getText()));
                    newTest.test1(textArea.getText(), textArea1.getText(), back);
                    setVisible(false);
                }
            });
            addShape.addActionListener(new ActionListener() {
                @Override
                public void actionPerformed(ActionEvent actionEvent) {
                    textArea.setText(textArea.getText() + (textArea.getText().length() != 0 ? "
" : "") + forShape.getText());
                    new VisualizeUI(textArea, textArea1, open);
                    dispose();
                }
            });
            delShape.addActionListener(new ActionListener() {
                @Override
                public void actionPerformed(ActionEvent actionEvent) {
```

```
                    textArea.setText(StringProcessor.stringDelShape(textArea.getText(),
forShape.getText()));
                    new VisualizeUI(textArea, textArea1, open);
                    dispose();
                }
            });
            forOpen.addActionListener(new ActionListener() {
                @Override
                public void actionPerformed(ActionEvent e) {
                    open.doClick();
                    dispose();
                }
            });
            panel.add(prevSt);
            panel.add(nextSt);
            panel.add(inTheEnd);
            panel.add(forShape);
            panel.add(addShape);
            panel.add(delShape);
            panel.add(scrollText);
            panel.add(scrollResult);
            panel.add(forOpen);
            add(panel);
            setLocationRelativeTo(null);
            setVisible(true);
        }
}
```

```
//CoordinateGenerator.java

package io.desmy.graph;

import io.desmy.project.*;
import java.util.HashMap;
public class CoordinateGenerator {
	private Point headPoint;
	private int diam;
	private HashMap<Node, Point> list;
	private Node head;
	private int offset = 50;
	private HashMap<Integer, Integer> off;
	public CoordinateGenerator(Bohr graphic) {
		off = new HashMap<>();
		list = new HashMap<>();
		headPoint = new Point(diam, diam);
		head = graphic.getHead();
		diam = 25;
		list.put(head, headPoint);
		off.put(1, 0);
		ForEach(head, 1);
	}
	private void ForEach(Node val, int level) {
		if(!off.containsKey(level+1))
			off.put(level+1, 0);
		val.getNextLevelNode().forEach((key, value) -> {
			int buf = off.get(level);
			Point v = new Point(headPoint.getX() + buf, headPoint.getY() +
offset*level);
			off.replace(level, buf+offset);
			list.put(value, v);
			ForEach(value, level + 1);
		});
	}
	public Node getH() { return head; }
	public Point getHead() { return headPoint; }
	public int getDiam() { return diam; }
	public HashMap<Node, Point> getList() { return list; }
}
```

Graph.java

```
package io.desmy.graph;

import io.desmy.project.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.HashMap;
public class Graph extends JPanel implements MouseListener, ActionListener,
MouseWheelListener {
	private CoordinateGenerator gen;
	private Bohr FH;
	private Point maxSize;
	public Graph(Bohr for_head) {
		gen = new CoordinateGenerator(for_head);
		maxSize = new Point(271,271);
		gen.getList().forEach((key, value) -> {
			if(value.getY() > maxSize.getY())
				maxSize.setY(value.getY());
			if(value.getX() > maxSize.getX())
				maxSize.setX(value.getX());
		});
		FH = for_head;
		addMouseListener(this);
		setFocusable(true);
		setPreferredSize(new
Dimension(maxSize.getX()+gen.getDiam(),maxSize.getY()+gen.getDiam()));
	}
	@Override
	public void paint(Graphics g) {
		int d = gen.getDiam();
		g.setColor(Color.white);
		g.fillRect(0,0, maxSize.getX()+gen.getDiam(),maxSize.getY()+gen.getDiam());
		//Every element
		gen.getList().forEach((key, value) -> {
			key.getNextLevelNode().forEach((keyN, valueN) -> {
				Point start = gen.getList().get(key);
				Point finish = gen.getList().get(valueN);
				g.setColor(Color.green);
				g.drawLine(start.getX() + d/2, start.getY() + d/2, finish.getX() + d/2,
finish.getY() + d/2);
				g.setColor(Color.black);
				g.drawString("" + keyN, (start.getX() + finish.getX() + d/2) / 2 + 1 ,
(start.getY() + finish.getY() + d) / 2 + 5);
			});});
		gen.getList().forEach((key, value) -> {
			g.setColor(Color.yellow);
			g.fillOval(value.getX(), value.getY(), gen.getDiam(), gen.getDiam());
			g.setColor(Color.black);
			g.drawOval(value.getX(), value.getY(), gen.getDiam(), gen.getDiam());
			if(key.getIndex() != -1)
				g.drawString("" + key.getIndex(), value.getX() + d/2 - 2, value.getY() +
d/2 + 5);
		});
		//Head
		g.setColor(Color.red);
		g.fillOval(gen.getHead().getX(), gen.getHead().getY(), gen.getDiam(),
gen.getDiam());
```

```
            //HeadBord
            g.setColor(Color.black);
            g.drawOval(gen.getHead().getX(), gen.getHead().getY(), gen.getDiam(),
gen.getDiam());
            Node now = FH.getNow();
            if(now != gen.getH()) {  //now - это указатель на текущий посещённый элемент
                  do {
                        g.setColor(Color.green);
                        HashMap<Node, Point> io = gen.getList();
                        g.fillOval(io.get(now).getX(), io.get(now).getY(), gen.getDiam(),
gen.getDiam());
                        //HeadBord
                        g.setColor(Color.black);
                        g.drawOval(io.get(now).getX(), io.get(now).getY(), gen.getDiam(),
gen.getDiam());
                        if (now.getIndex() != -1)
                              g.drawString("" + now.getIndex(), io.get(now).getX() + d / 2 - 2,
io.get(now).getY() + d / 2 + 5);
                        now = now.suffixLink;
                  } while(now != gen.getH());
            }
            now = FH.getNow();
            g.setColor(Color.red);
            g.drawOval(gen.getList().get(now).getX(), gen.getList().get(now).getY(),
gen.getDiam(), gen.getDiam());
      }
      public void rePaint() { repaint(); }
      @Override
      public void actionPerformed(ActionEvent e) {}
      @Override
      public void mouseClicked(MouseEvent e) {}
      @Override
      public void mousePressed(MouseEvent e) {}
      @Override
      public void mouseReleased(MouseEvent e) {}
      @Override
      public void mouseEntered(MouseEvent e) {}
      @Override
      public void mouseExited(MouseEvent e) {}
      @Override
      public void mouseWheelMoved(MouseWheelEvent e) {}
}
```

Point.java

```
package io.desmy.graph;

public class Point {
	private int pointX;
	private int pointY;
	public Point(int X, int Y) {
		pointX = X;
		pointY = Y;
	}
	public int getX() {
		return pointX;
	}
	public int getY() {
		return pointY;
	}
	public void setX(int X) { pointX = X; }
	public void setY(int Y) { pointY = Y; }
}
```