Name of the Course : Beginning Java Data Structures and Algorithms

Module : String Matching Algorithms

Tool Stack : Java8 and Junit5

Problem Statement : Provide a code solution to search string pattern pattern using BoyerMoore algorithm implementation.

Description : Create three classes, one BoyerMoore class with text as String and pattern as String fields and with a parameterized constructor,

One BoyerMooreBO class with six static methods

1. public static int findPattern(BoyerMoore boyerMoore), which accepts BoyerMoore object and returns an integer value

2. public static int indexOf(char[] text, char[] pattern), which accepts char[], char[] i.e two char arrays and returns an int value

3. private static int[] makeCharTable(char[] pattern), which accepts char[][] i.e one char arrays and returns an int value

4. private static int[] makeOffsetTable(char[] pattern), which accepts char[][] i.e one char arrays and returns an int value

5. private static boolean isPrefix(char[] pattern, int p), which accepts char[], int i.e one char array and one int value and returns true or false

6. private static int suffixLength(char[] pattern, int p), which accepts char[], int i.e one char array and one int value and returns an int value

and one MainClass with one static method

1. public static void main method, for reading the text and pattern from input devices and call the findPattern method to test it.

Code:

**package** yaksha;

**public** **class** BoyerMoore {

**private** String text;

**private** String pattern;

**public** BoyerMoore(String text, String pattern) {

**super**();

**this**.text = text;

**this**.pattern = pattern;

}

**public** String getText() {

**return** text;

}

**public** **void** setText(String text) {

**this**.text = text;

}

**public** String getPattern() {

**return** pattern;

}

**public** **void** setPattern(String pattern) {

**this**.pattern = pattern;

}

}

**package** yaksha;

**public** **class** BoyerMooreBO {

// Method to findPattern

**public** **static** **int** findPattern(BoyerMoore boyerMoore) {

String text1 = boyerMoore.getText();

String pattern1 = boyerMoore.getPattern();

**char**[] text = text1.toCharArray();

**char**[] pattern = pattern1.toCharArray();

**int** pos = *indexOf*(text, pattern);

**return** pos;

}

// Method to calculate index of pattern substring

**public** **static** **int** indexOf(**char**[] text, **char**[] pattern) {

**if** (pattern.length == 0)

**return** 0;

**int** charTable[] = *makeCharTable*(pattern);

**int** offsetTable[] = *makeOffsetTable*(pattern);

**for** (**int** i = pattern.length - 1, j; i < text.length;) {

**for** (j = pattern.length - 1; pattern[j] == text[i]; --i, --j)

**if** (j == 0)

**return** i;

i += Math.*max*(offsetTable[pattern.length - 1 - j], charTable[text[i]]);

}

**return** -1;

}

// Method to make the jump table based on the mismatched character information

**private** **static** **int**[] makeCharTable(**char**[] pattern) {

**final** **int** ALPHABET\_SIZE = 256;

**int**[] table = **new** **int**[ALPHABET\_SIZE];

**for** (**int** i = 0; i < table.length; ++i)

table[i] = pattern.length;

**for** (**int** i = 0; i < pattern.length - 1; ++i)

table[pattern[i]] = pattern.length - 1 - i;

**return** table;

}

// Method to make the jump table based on the scan offset which mismatch occurs.

**private** **static** **int**[] makeOffsetTable(**char**[] pattern) {

**int**[] table = **new** **int**[pattern.length];

**int** lastPrefixPosition = pattern.length;

**for** (**int** i = pattern.length - 1; i >= 0; --i) {

**if** (*isPrefix*(pattern, i + 1))

lastPrefixPosition = i + 1;

table[pattern.length - 1 - i] = lastPrefixPosition - i + pattern.length - 1;

}

**for** (**int** i = 0; i < pattern.length - 1; ++i) {

**int** slen = *suffixLength*(pattern, i);

table[slen] = pattern.length - 1 - i + slen;

}

**return** table;

}

// Method to check if needle[p:end] a prefix of pattern

**private** **static** **boolean** isPrefix(**char**[] pattern, **int** p) {

**for** (**int** i = p, j = 0; i < pattern.length; ++i, ++j)

**if** (pattern[i] != pattern[j])

**return** **false**;

**return** **true**;

}

// Method to returns the maximum length of the substring ends at p and is a suffix

**private** **static** **int** suffixLength(**char**[] pattern, **int** p) {

**int** len = 0;

**for** (**int** i = p, j = pattern.length - 1; i >= 0 && pattern[i] == pattern[j]; --i, --j)

len += 1;

**return** len;

}

}

**package** yaksha;

**import** java.io.BufferedReader;

**import** java.io.IOException;

**import** java.io.InputStreamReader;

**public** **class** MainClass {

**public** **static** **void** main(String[] args) **throws** IOException {

BufferedReader br = **new** BufferedReader(**new** InputStreamReader(System.***in***));

System.***out***.println("Boyer Moore Algorithm Test\n");

System.***out***.println("\nEnter Text\n");

String text = br.readLine();

System.***out***.println("\nEnter Pattern\n");

String pattern = br.readLine();

BoyerMoore boyerMoore = **new** BoyerMoore(text, pattern);

**int** pos = BoyerMooreBO.*findPattern*(boyerMoore);

**if** (pos == -1)

System.***out***.println("\nNo Match\n");

**else**

System.***out***.println("Pattern found at position : " + pos);

}

}

Junit Testing

**package** yaksha;

**import** **static** yaksha.TestUtils.*businessTestFile*;

**import** **static** yaksha.TestUtils.*currentTest*;

**import** **static** yaksha.TestUtils.*yakshaAssert*;

**import** org.junit.jupiter.api.Test;

**class** MainClassTest {

@Test

**public** **void** testExceptionConditon() **throws** Exception {

TestUtils.*yakshaAssert*(TestUtils.*currentTest*(), **true**, TestUtils.*exceptionTestFile*);

}

@Test

**public** **void** testBoundaryCondition() **throws** Exception {

TestUtils.*yakshaAssert*(TestUtils.*currentTest*(), **true**, TestUtils.*boundaryTestFile*);

}

@Test

**void** testFindPattern() **throws** Exception {

// Test will pass

String text = "abcdefghijklmnopqrstuvwxyz";

String pattern = "qrstuv";

String receivedResult = **null**;

String expectedResult = "Pattern found at position : 16";

BoyerMoore boyerMoore = **new** BoyerMoore(text, pattern);

**int** pos = BoyerMooreBO.*findPattern*(boyerMoore);

**if** (pos == -1) {

receivedResult = "No Match";

} **else** {

receivedResult = "Pattern found at position : " + pos;

}

*yakshaAssert*(*currentTest*(), (expectedResult.equals(receivedResult) ? "true" : "false"), *businessTestFile*);

}

}

pom.xml

<project xmlns=*"http://maven.apache.org/POM/4.0.0"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>iiht.yaksha.bmss</groupId>

<artifactId>BoyerMooreBegJvDSnAlgoStrMtchAlgo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>BoyerMooreBegJvDSnAlgoStrMtchAlgo</name>

<description>BoyerMooreBegJvDSnAlgoStrMtchAlgo</description>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<maven.compiler.source>1.8</maven.compiler.source>

<maven.compiler.target>${maven.compiler.source}</maven.compiler.target>

<junit.jupiter.version>5.5.2</junit.jupiter.version>

<junit.platform.version>1.5.2</junit.platform.version>

</properties>

<dependencies>

<!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<version>1.18.12</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter-engine</artifactId>

<version>${junit.jupiter.version}</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.junit.platform</groupId>

<artifactId>junit-platform-runner</artifactId>

<version>${junit.platform.version}</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>8.0.11</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.8.1</version>

</plugin>

<plugin>

<artifactId>maven-surefire-plugin</artifactId>

<version>2.22.2</version>

</plugin>

</plugins>

</build>

</project>

Test Data1

Boyer Moore Algorithm Implementation

Enter Text :

abcdefghijklmnopqrstuvwxyz

Enter Pattern :

qrstuv

Pattern found at position : 16

Learning outcome: Participant could able to learn string matching using BoyerMoore algorithm implementation.