/\*Matthew Irvin

CS201 assignment 2 Project - Bad Character

Jan 25, 2025

\*/

import java.util.Scanner;

/\*\*

\* This program allows users to search for a pattern in the names of the 50 US states

\* using the Boyer-Moore algorithm's bad character rule. It has a menu-driven interface

\* for displaying all states, searching a pattern, and exits program.

\*/

public class App {

// Array containing all 50 US states

private static final String[] STATES = {

"Alabama", "Alaska", "Arizona", "Arkansas", "California", "Colorado", "Connecticut", "Delaware", "Florida", "Georgia",

"Hawaii", "Idaho", "Illinois", "Indiana", "Iowa", "Kansas", "Kentucky", "Louisiana", "Maine", "Maryland",

"Massachusetts", "Michigan", "Minnesota", "Mississippi", "Missouri", "Montana", "Nebraska", "Nevada", "New Hampshire", "New Jersey",

"New Mexico", "New York", "North Carolina", "North Dakota", "Ohio", "Oklahoma", "Oregon", "Pennsylvania", "Rhode Island", "South Carolina",

"South Dakota", "Tennessee", "Texas", "Utah", "Vermont", "Virginia", "Washington", "West Virginia", "Wisconsin", "Wyoming"

};

public static void main(String[] args) {

Scanner inputReader = new Scanner(System.in);

int userChoice;

// Main program loop

do {

displayMenuOptions();

if (inputReader.hasNextInt()) {

userChoice = inputReader.nextInt();

inputReader.nextLine(); // Consume leftover newline

} else {

System.out.println("Invalid input. Please enter a number.");

inputReader.nextLine(); // Clear invalid input

userChoice = -1; // Forces loop to continue

}

// Process user's choice

switch (userChoice) {

case 1:

showAllStates();

break;

case 2:

performStateSearch(inputReader);

break;

case 3:

System.out.println("Exiting. BYE!");

break;

default:

System.out.println("Invalid choice. Please try again.");

}

} while (userChoice != 3);

inputReader.close();

}

// Display menu options

private static void displayMenuOptions() {

System.out.println("\nPlease select an option:");

System.out.println("1) Display the text");

System.out.println("2) Search");

System.out.println("3) Exit program");

System.out.print("Enter choice ");

}

// Display all states in the STATES array

private static void showAllStates() {

System.out.println("\nHere are the 50 US states:");

for (String state : STATES) {

System.out.println(state);

}

}

/\*\*

\* Performs search for the user's given pattern in the names of the 50 US states.

\* Uses the Boyer-Moore algorithm's bad character rule to find matches.

\*

\* Scanner object used to read user input.

\*/

private static void performStateSearch(Scanner scanner) {

System.out.print("Enter a part of the name of a state to search for: ");

String pattern = scanner.nextLine();

System.out.println("Searching for \"" + pattern + "\"...");

boolean found = false;

for (int i = 0; i < STATES.length; i++) {

String state = STATES[i];

int index = modifiedBoyerMooreSearch(state.toLowerCase(), pattern.toLowerCase());

if (index != -1) {

System.out.println("Found in state #" + (i + 1) + ": \"" + state + "\" at index " + index);

found = true;

}

}

if (!found) {

System.out.println("No matches found.");

}

}

/\*\*

\* Searches for a pattern within a given text using the bad character rule

\* from the Boyer-Moore algorithm. This method efficiently locates the starting

\* position of the pattern in the text or returns -1 if no match is found.

\*

\* @param text We use this string to search for the pattern.

\* @param pattern The substring to locate within the text.

\* @return The index where the pattern begins in the text, or -1 if it is not present.

\*/

private static int modifiedBoyerMooreSearch(String text, String pattern) {

int[] lastOccurrence = new int[256];

for (int i = 0; i < 256; i++) {

lastOccurrence[i] = -1;

}

// Preprocess the pattern

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

lastOccurrence[pattern.charAt(i)] = i;

}

int i = 0; // Index for text

while (i <= text.length() - pattern.length()) {

int j = pattern.length() - 1; // Index for pattern

// Compare characters from right to left

while (j >= 0 && pattern.charAt(j) == text.charAt(i + j)) {

j--;

}

if (j < 0) {

// Pattern found

return i;

} else {

// Shift the pattern

i += Math.max(1, j - lastOccurrence[text.charAt(i + j)]);

}

}

// Pattern not found

return -1;

}

}