University of Windsor

Master of Applied Computing

COMP 8547 Advanced Computing Concepts – Summer 2024

Assignment 3

Task 5: Page Ranking Using Frequency Count and Boyer-Moore Algorithm

Technical Report

Author:
Yesha Umeshkumar Patel
110164042

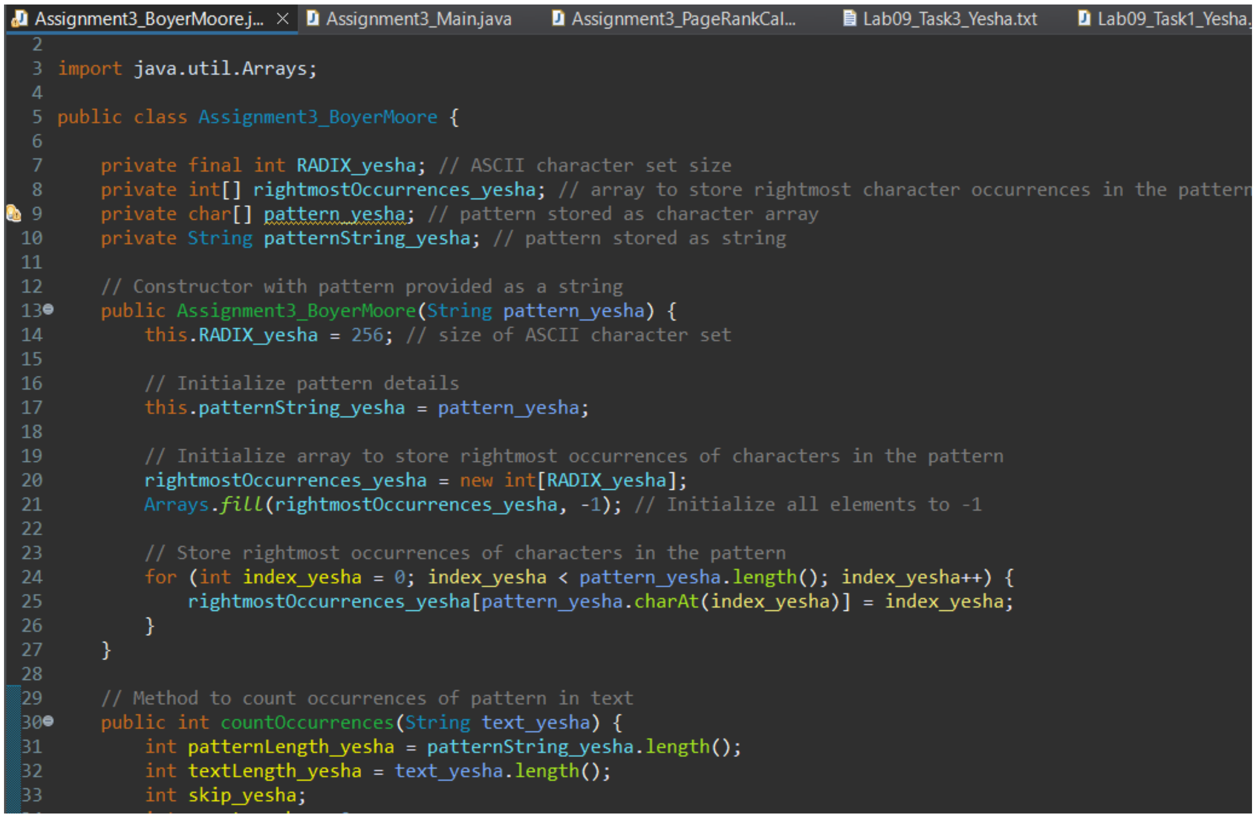
Submitted to:
Dr. Olena Syrotkina

Introduction

The task involves developing a page ranking system that utilizes the Boyer-Moore algorithm to efficiently count keyword occurrences in multiple web pages. The goal is to rank these pages based on the frequency of these keywords, providing a sorted list that prioritizes pages most relevant to the search criteria.

Use the Boyer-Moore algorithm to count the occurrences of the keyword in each web page.

The `Assignment3_BoyerMoore` class implements the Boyer-Moore algorithm as shown in Figure 1. It takes a keyword string as input and provides a method to count its occurrences within another text string. This algorithm offers efficient searching by utilizing character mismatches to skip unnecessary comparisons.



Assignment3_BoyerMoore.j... × | Assignment3_Main.java | Assignment3_PageRankCal... | Lab09_Task3_Yesha.txt | Lab09_Task1_Yesha.

```java

import java.util.Arrays;

public class Assignment3_BoyerMoore {

    private final int RADIX_yesha; // ASCII character set size
    private int[] rightmostOccurrences_yesha; // array to store rightmost character occurrences in the pattern
    private char[] pattern_yesha; // pattern stored as character array
    private String patternString_yesha; // pattern stored as string

    // Constructor with pattern provided as a string
    public Assignment3_BoyerMoore(String pattern_yesha) {
        this.RADIX_yesha = 256; // size of ASCII character set

        // Initialize pattern details
        this.patternString_yesha = pattern_yesha;

        // Initialize array to store rightmost occurrences of characters in the pattern
        rightmostOccurrences_yesha = new int[RADIX_yesha];
        Arrays.fill(rightmostOccurrences_yesha, -1); // Initialize all elements to -1

        // Store rightmost occurrences of characters in the pattern
        for (int index_yesha = 0; index_yesha < pattern_yesha.length(); index_yesha++) {
            rightmostOccurrences_yesha[pattern_yesha.charAt(index_yesha)] = index_yesha;
        }
    }

    // Method to count occurrences of pattern in text
    public int countOccurrences(String text_yesha) {
        int patternLength_yesha = patternString_yesha.length();
        int textLength_yesha = text_yesha.length();
        int skip_yesha;
```

Figure 1: Assignment3_BoyerMoore

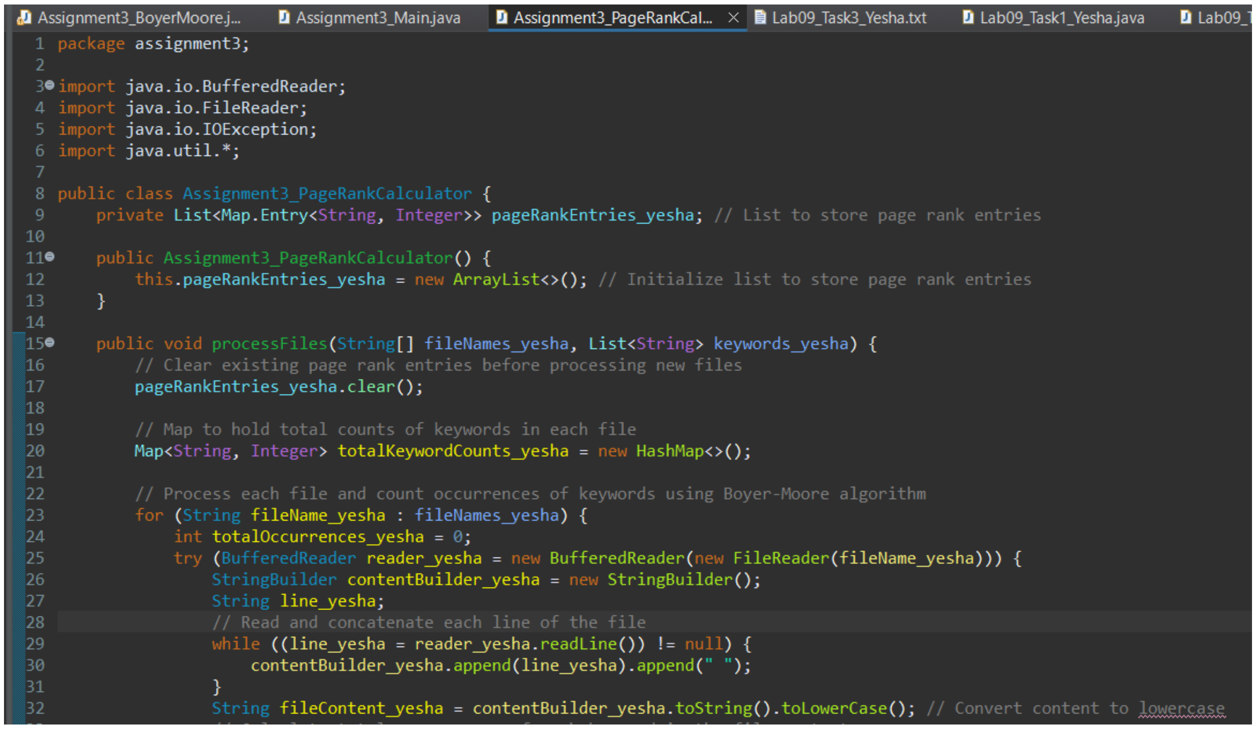
Rank the pages based on the frequency count.

The `Assignment3_PageRankCalculator` class takes a list of file names (representing web pages) and keywords as shown in Figure 2. It iterates through each file:

- Reads the content of the file.
- Uses the `Assignment3_BoyerMoore` class to count the occurrences of each keyword in the file content.
- Stores the keyword and its frequency count for the current file.

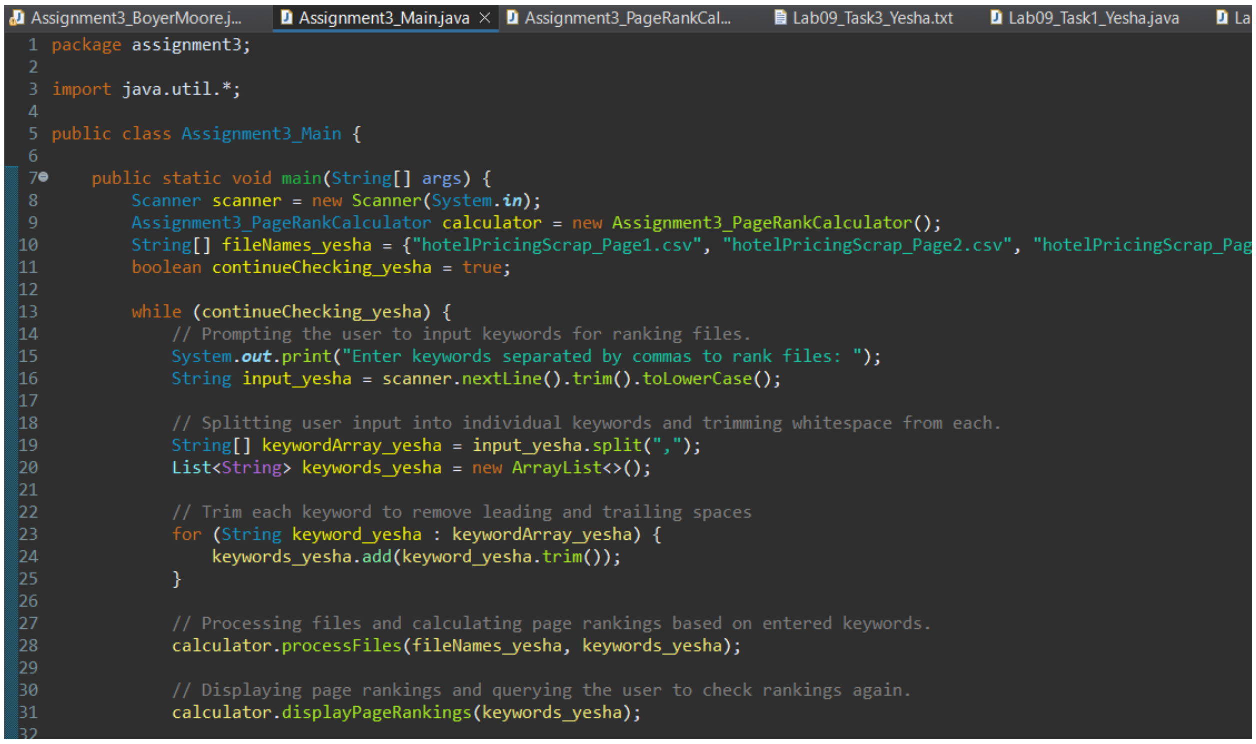
The `Assignment3_Main` class manages user interaction and program flow as shown in Figure 3:

- Prompts the user to enter keywords separated by commas.
- Processes the input and creates a list of keywords.
- Creates an instance of `Assignment3_PageRankCalculator` and calls its methods to process files and calculate rankings.
- Displays the ranked list of files based on the keyword frequencies.
- Asks the user if they want to analyze another set of keywords



```java
package assignment3;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.*;

public class Assignment3_PageRankCalculator {
    private List<Map.Entry<String, Integer>> pageRankEntries_yesha; // List to store page rank entries

    public Assignment3_PageRankCalculator() {
        this.pageRankEntries_yesha = new ArrayList<>(); // Initialize list to store page rank entries
    }

    public void processFiles(String[] fileNames_yesha, List<String> keywords_yesha) {
        // Clear existing page rank entries before processing new files
        pageRankEntries_yesha.clear();

        // Map to hold total counts of keywords in each file
        Map<String, Integer> totalKeywordCounts_yesha = new HashMap<>();

        // Process each file and count occurrences of keywords using Boyer-Moore algorithm
        for (String fileName_yesha : fileNames_yesha) {
            int totalOccurrences_yesha = 0;
            try (BufferedReader reader_yesha = new BufferedReader(new FileReader(fileName_yesha))) {
                StringBuilder contentBuilder_yesha = new StringBuilder();
                String line_yesha;
                // Read and concatenate each line of the file
                while ((line_yesha = reader_yesha.readLine()) != null) {
                    contentBuilder_yesha.append(line_yesha).append(" ");
                }
                String fileContent_yesha = contentBuilder_yesha.toString().toLowerCase(); // Convert content to lowercase
```

Figure 2: Assignment3_PageRankCalculator.java

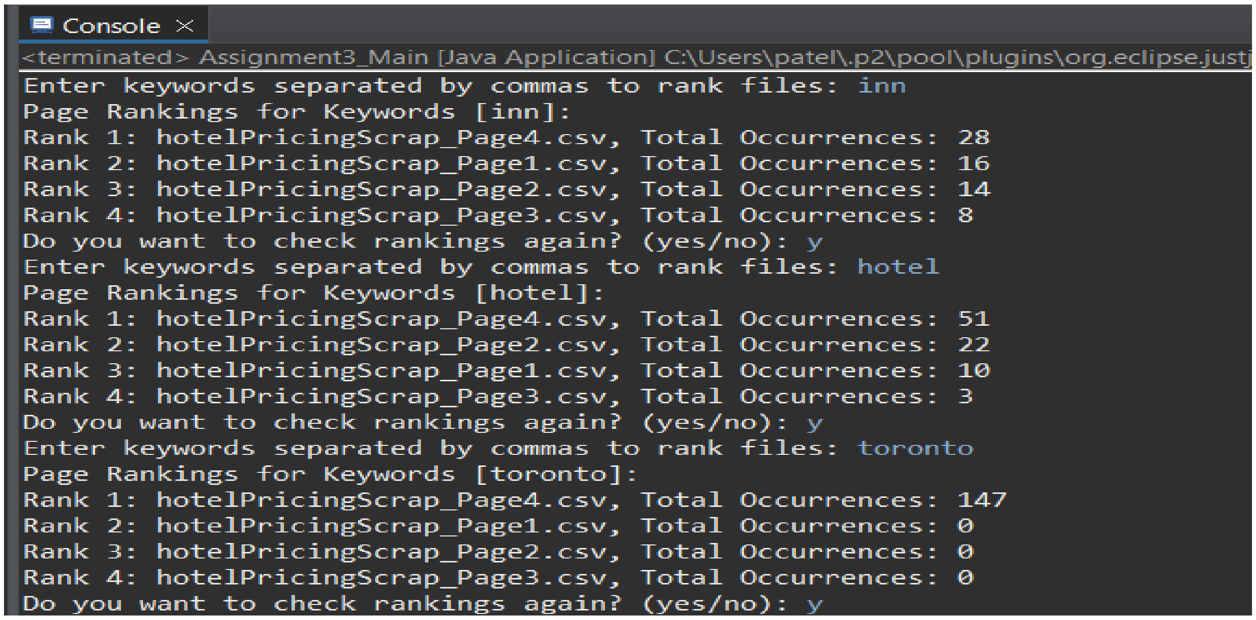


Assignment3_BoyerMoore.j... Assignment3_Main.java Assignment3_PageRankCal... Lab09_Task3_Yesha.txt Lab09_Task1_Yesha.java La

```
package assignment3;

import java.util.*;

public class Assignment3_Main {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Assignment3_PageRankCalculator calculator = new Assignment3_PageRankCalculator();
        String[] fileNames_yesha = {"hotelPricingScrap_Page1.csv", "hotelPricingScrap_Page2.csv", "hotelPricingScrap_Pag
        boolean continueChecking_yesha = true;

        while (continueChecking_yesha) {
            // Prompting the user to input keywords for ranking files.
            System.out.print("Enter keywords separated by commas to rank files: ");
            String input_yesha = scanner.nextLine().trim().toLowerCase();

            // Splitting user input into individual keywords and trimming whitespace from each.
            String[] keywordArray_yesha = input_yesha.split(",");
            List<String> keywords_yesha = new ArrayList<>();

            // Trim each keyword to remove leading and trailing spaces
            for (String keyword_yesha : keywordArray_yesha) {
                keywords_yesha.add(keyword_yesha.trim());
            }

            // Processing files and calculating page rankings based on entered keywords.
            calculator.processFiles(fileNames_yesha, keywords_yesha);

            // Displaying page rankings and querying the user to check rankings again.
            calculator.displayPageRankings(keywords_yesha);
```

Figure 3: Assignment3_Main.java

Output



```
Console
<terminated> Assignment3_Main [Java Application] C:\Users\patel\.p2\pool\plugins\org.eclipse.justj
Enter keywords separated by commas to rank files: inn
Page Rankings for Keywords [inn]:
Rank 1: hotelPricingScrap_Page4.csv, Total Occurrences: 28
Rank 2: hotelPricingScrap_Page1.csv, Total Occurrences: 16
Rank 3: hotelPricingScrap_Page2.csv, Total Occurrences: 14
Rank 4: hotelPricingScrap_Page3.csv, Total Occurrences: 8
Do you want to check rankings again? (yes/no): y
Enter keywords separated by commas to rank files: hotel
Page Rankings for Keywords [hotel]:
Rank 1: hotelPricingScrap_Page4.csv, Total Occurrences: 51
Rank 2: hotelPricingScrap_Page2.csv, Total Occurrences: 22
Rank 3: hotelPricingScrap_Page1.csv, Total Occurrences: 10
Rank 4: hotelPricingScrap_Page3.csv, Total Occurrences: 3
Do you want to check rankings again? (yes/no): y
Enter keywords separated by commas to rank files: toronto
Page Rankings for Keywords [toronto]:
Rank 1: hotelPricingScrap_Page4.csv, Total Occurrences: 147
Rank 2: hotelPricingScrap_Page1.csv, Total Occurrences: 0
Rank 3: hotelPricingScrap_Page2.csv, Total Occurrences: 0
Rank 4: hotelPricingScrap_Page3.csv, Total Occurrences: 0
Do you want to check rankings again? (yes/no): y
```

Figure 4: Output

- Each time the program prompts the user to enter keywords separated by commas. These keywords are used to rank the files based on the total occurrences of these keywords in each file as shown in Figure 4.
- For each set of keywords entered, the program calculates how many times each keyword appears in each file using the Boyer-Moore algorithm. It then ranks the files from highest to lowest total occurrences of the keywords.
- The program asks if the user wants to continue checking rankings (yes or y) or stop (no or n). This loop continues until the user decides to stop.

Conclusion

This program demonstrates a simple page ranking approach using keyword frequency and the Boyer-Moore algorithm. The Boyer-Moore algorithm showcases a technique for speeding up search operations within large text files. This program helped me in understanding more complex ranking systems.