Module 12 Assignment: Pattern Matching

Marcar esta página

Coding Assignment fecha límite Nov 4, 2023 16:01 -05

Pattern Matching

For this assignment you will be implementing the **Boyer-Moore** pattern matching algorithm. You should find **all** occurrences of the pattern in the text, not just the first match. The occurrences are returned as a list of integers; the list should contain the indices of occurrences in ascending order. There is information about Boyer-Moore in the javadocs with additional implementation details below.

Be sure that you check the simple failure cases as soon as possible. For example, if the pattern is longer than the text, then do not do any preprocessing on the pattern/text and just return an empty list since there cannot be any occurrences of the pattern in the text.

Note that for pattern matching, we refer to the text length as n and the pattern length as m.

IMPORTANT:

- **You will be given 5 attempts on this assignment, with a 30 minute cooldown between submissions.**
- **Please run your code before each submission to ensure that there are no formatting errors! If there are formatting errors in your code, your code will not be graded and a submission attempt will be logged. For more information, please review the Vocareum overview below.**

CharacterComparator

CharacterComparator is a comparator that takes in two characters and compares them. This allows you to see how many times you have called compare(). Besides this functionality, its return values are what you'd expect a properly implemented compare() method to return. You **must** use this comparator as the number of times you call compare() with it will be used when testing your assignment.

If you do not use the passed in comparator, this will cause tests to fail and will significantly lower your grade on this assignment.

You must implement the algorithms as they were taught in class. We are expecting **exact** comparison counts for this assignment. If you are getting fewer comparison counts than expected, it means one of two things: either you implemented the algorithm wrong (most likely) or you are using an optimization not taught in the class (unlikely).

Boyer-Moore

Last Occurrence Table

The Boyer-Moore algorithm relies on preprocessing the pattern. Before actually searching, the algorithm generates a last occurrence table. The table allows the algorithm to skip sections of the text, resulting in more

generates a last occurrence table. The table allows the algorithm to skip sections of the text, resulting in more efficient string searching. The last occurrence table should be a mapping from each character in the alphabet (the set of all characters that may be in the pattern or the text) to the last index the character appears in the pattern. If the character is not in the pattern, then -1 is used as the value, though you should not explicitly add all characters that are not in the pattern into the table.

Searching Algorithm

Key properties of Boyer-Moore include matching characters starting at the end of the pattern, rather than the beginning, and skipping along the text in jumps of multiple characters rather than searching every single character in the text.

The shifting rule considers the character in the text at which the comparison process failed (assuming that a failure occurred). If the last occurrence of that character is to the left in the pattern, shift so that the pattern occurrence aligns with the mismatched text occurrence. If the last occurrence of the mismatched character does not occur to the left in the pattern, shift the pattern over by one (to prevent the pattern from moving backwards). In addition, if the mismatched character does not exist in the pattern at all (no value in last table) then pattern shifts completely past this point in the text.

For finding multiple occurrences, if you find a match, shift the pattern over by one and continue searching.

General Tips

- The **getOrDefault()** method from Java's Map will be useful for retrieving the last occurrence value of a character not in the pattern.
- How can you tell if the pattern has gone past the text? Is there any significance in the value of $n - m$?
- We highly recommend copying the starter code and working in your preferred IDE in order to have access to features such as code completion, auto-formatting, and much more!

Here are general assignment guidelines that should be followed.

- Do not include any package declarations in your classes.
- Do not change any existing class headers, constructors, instance/global variables, or method signatures. For example, do not add throws to the method headers since they are not necessary. Instead, exceptions should be thrown as follows: `throw new InsertExceptionHere("Error: some exception was thrown");`
- All helper methods you choose to write should be made private. Recall the idea of Encapsulation in Object-Oriented Programming!
- Do not use anything that would trivialize the assignment. (e.g. Don't import/use java.util.ArrayList for an ArrayList assignment.)
- Always be very conscious of efficiency. Even if your method is to be $O(n)$, traversing the structure multiple times is considered inefficient unless that is absolutely required (and that case is extremely rare).
- If applicable, use the generic type of the class; do not use the raw type of the class. For example, use `new LinkedList<Integer>()` instead of `new LinkedList()`.

Use of the following statements should be avoided at all times.

package	System.arraycopy()	clone()
assert()	Arrays class	Array class
Thread class	Collections class	Collection.toArray()
Reflection APIs	Inner or nested classes	Lambda Expressions

The Vocareum (code editor) interface has six main components:

- The **Drop-Down** in the top left. This lets you choose from multiple available files. Note that this drop-down will only be visible in assignments that require multiple files.
- The **Run** button. This will compile your code and run a file scan. Running your code will not count towards your total allowed submission attempts, therefore you are free to run as many times as needed.
- The **Submit** button. This will compile your code, run a file scan, grade your assignment, and output results to console. Note that for most assignments in this class, you will only be allowed a limited number of submissions. A

submission is counted when the submit button is clicked, regardless of whether or not your code can compile or if there are any file issues. Therefore, we **highly recommend** that you run your code before submitting to ensure that there are no issues that will prevent your code from being graded and that every submission attempt will generate meaningful results.

- The **Reset** button. This will revert all your changes and reset your code to the default code template.
- The **Code Window**. This is where you will write your code. For large coding assignments, we highly recommend copying the starter code and working in your preferred IDE to have access to features such as code completion, auto-formatting, and much more!
- The **Output Window**. This window will appear whenever you run or submit your code and will display the output for you to view.

For additional help, please visit the Vocareum information page located in the course information module!

‹ Anterior Siguiente ›