CS2010: Data Structure and Algorithms - HT: Assignment 2

In this assignment you will implement several String processing algorithms and use them in two applications: (i) bus service network real-time information, and (ii) English-language dictionary.

Total points for this assignment: 200 (100 automatic, 100 marked by demonstrators)

The following will be marked:

1. Correctness of your results, JUnit tests, and test code coverage – automatic mark through web-cat, 100 points
2. Correct and efficient implementation of your algorithms (eg your string search method uses kmp algorithm as required rather than brute force etc) and your answers to questions match – marked by demonstrators, 100 points

Submission and automatic marking is through https://webcat.scss.tcd.ie/cs2012/WebObjects/Web-CAT.woa. Submission of final version both through Web-CAT and Blackboard.

Deadline: March 8th 2018 23:45

Late assignments will be deducted 40 points per day

Please submit only KMPSearch.java, KMPSearchTest.java, TST.java, and TSTTest.java in a single zip file. Do not submit input or jar files.

Assignment specification

1. KMP Search

Download KMPSearch.java and KMPSearchTest.java.

A - Write a java class KMPSearch in KMPSearch.java file (please do not use custom packages as web-cat will give an error) which should implement at least the following methods, using KMP algorithm:

- public boolean contains (String txt, String pat) – a method which checks whether pattern pat occurs at least once in String txt, returns true if it is, false otherwise
- public int searchFirst (String txt, String pat) – a method which returns the index of the first occurrence of a pattern pat in String txt. Returns -1 if the pattern is not present
- public int searchAll (String txt, String pat) – a method which returns the total number of all non-overlapping occurrences of a pattern pat in the String txt

B - Write a java class KMPSearchTest in KMPSearchTest.java file, which should implement JUnit tests for your KMPSearch class.

Your goal is to write enough tests so that:

- Each method in in the classes is tested at least once,
- Each decision (that is, every branch of if-then-else, for, and other kinds of choices) in is tested at least once,
- Each line of code in both classes is executed at least once from the tests.

The submission server will analyse your tests and code to determine if the above criteria have been satisfied.

C – Add a main method to KMPSearchTest.java and use it to answer the questions below.

Read the full content of the file BUSES_SERVICE_0.json in as a String (The file has been obtained by using Vancouver bus operator TransLink developer API, available on https://developer.translink.ca and it contains real-time bus information). Answer the following questions, by executing methods from your KMP class:

1. How many total vehicles is there information on in the BUSES_SERVICE_0.json file, i.e., how many occurrences of the pattern "VehicleNo" is there in the file?

2. Does the file contain information about the vehicle number 16555, i.e., is the pattern "16555" present in the file?

3. Locate the first record about a bus heading to HAMPTON PARK, i.e., return the index of the first occurrence of the pattern "HAMPTON PARK" in the file?

4. Does the file contain information about the vehicle number 9043409?

2. Ternary Search Trie (TST)

Download TST.java and TSTTest.java files.

A - Write a java class TST in TST.java file, which should implement at least the following methods:

- public int size() – returns the number of values stored in the trie
- public boolean contains(String key) – returns true if the trie contains the key
- public Value get(String key) – returns the value stored in the node with the given key
- public void put(String key, Value val) – stores the Value val in the node with the given key
- public LinkedList<String> keysWithPrefix(String prefix) - returns the linked list containing all the keys present in the trie that start with the prefix passed as a parameter, sorted in the alphabetical order. You are allowed use java.util.LinkedList implementation

B - Write a java class TSTTest in TSTTest.java file, which should implement JUnit tests for your TST class.

Your goal is to write enough tests so that:

- Each method in in the classes is tested at least once,
- Each decision (that is, every branch of if-then-else, for, and other kinds of choices) in is tested at least once,
- Each line of code in both classes is executed at least once from the tests.
  The submission server will analyse your tests and code to determine if the above criteria have been satisfied.

C - Add a main method to TSTTest.java and use it to answer the questions below.

As you might have noticed in KMP part of the assignment, BUSES_SERVICE_0.json is in .json format. JSON stands for JavaScript Object Notation, and is a lightweight data-interchange format, often used for exchanging information between a browser and a server. Using JSON libraries, we can easily parse JSON files to read individual records/fields from it, rather than processing it as a single string like we did in the part 1. To do this, you will need to download a .jar file containing JSON

Simple org.json.simple library from the blackboard assignment page (please use this library rather than one of the other JSON processing libraries, as that is the one we have set up the web-cat with). Documentation for the library is available at http://alex-public-doc.s3.amazonaws.com/json_simple-1.1/index.html.

Read in all the bus records and store the subset of that information in a TST as follows: use Destination as a key, and in the value field store the total number of records present in the file relating to that destination (ignore all the other fields). Therefore, when adding a destination node, if that key is not already present in the trie, add it with a count value of 1; if the key is already present, increase the current value count by 1, to indicate you have encountered another record about that destination. Answer the following questions, by executing methods from your TST class:

1. How many unique destinations is there in the file?
2. Is there a bus going to the destination "SOUTHSIDE"?
3. How many records is there about the buses going to the destination beginning with the pattern "DOWN"

D - In this part, you will use your TST class to work with a much larger trie than the one in part C, so proceed to this part only when you are sure your trie is working correctly. Download the file google-books-common-words.txt (available on the Blackboard assignment page, originally obtained from Peter Norvig's website on http://norvig.com/google-books-common-words.txt). The file contains ~100k words that most commonly occur in the books archived in Google Books. Read in the words in a trie, with the word as a key, and the frequency stored as a value (Note that some of the frequencies are too large to fit into an integer so please use long data type).

In the main method in TSTTest.java, write the code using TST to answer the following questions:

4. How many words is there present in the file?
5. What is the frequency of the word "ALGORITHM?"
6. Is the word "EMOJI" present in the list of most common Google Books words?
7. Is the word "BLAH" present in the list of most common Google Books words?
8. How many words is there in the file starting with "TEST"?

For fun – not marked: 100k words that most commonly occur in the books archived in Google Books effectively constitute a version of an English dictionary. You can use this file and the TST you have implemented to spellcheck any input, i.e., check if the word encountered is present in the dictionary, and potentially offer suggestions for correct spelling by returning words obtainable by adding any character to the beginning or end of the inputted string, removing any single character from the inputted string or swapping any two adjacent characters in the string.

Appendix: reminder of general assignment instructions from Semester 1

Please see a walkthrough on how to submit an assignment on Web-CAT.

When you upload code for an assignment to the submission server, it compiles it and runs your JUnit tests, giving you back an automatic score. This score is part of your marks for this assignment; the other part is given manually by the teaching staff. You can improve and reupload your code to improve your score. The only limit is the submission deadline.

For security reasons the submission server is only accessible from the campus network. If you want to access it from home you need to connect to the campus network via a Virtual Private Network (VPN) with your SCSS account. Instructions.

Students are allowed to discuss assignments but not to share code! Sharing code will result in reduced marks for all students involved and the consequences described in the College rules. If you discuss an assignment with fellow students then you must write the names of the students in your submission. All students must complete the College's online seminar about plagiarism before submitting any assignment.

If you are having trouble with assignments then you can attend both lab hours, which will give you more contact hours with teaching staff. If you are seeking support with more basic Java programming you can additionally attend the Undergraduate Programming Centre.

- Write your name next to @author at the beginning of each file
- Write the names of people you discussed this assignment with under the @author line. Do not share code and do not write code for others!
- You need to adequately test each method in your source code by adding sufficient jUnit tests
- Do not import data structures from the java libraries unless specified you are allowed do so.
- The submission server for this assignment will open shortly.