ISTE-222 Assignment 2: Non-Linear Data Structures

A Tree of Words

Problem Statement:

The purpose of this assignment is to apply and extend your knowledge of non-linear data structures to a relatively complicated scenario by writing your own methods to build a binary search tree containing objects and choosing an additional data structure to perform a subtask of the problem. **You may only use data structures that we've learned about in class (no ArrayList)**.

Written communication dates back several thousand years. Literary works have captivated us and brought out some of the very best in humanity. In this assignment, you'll be working with a collection of quotes (see quotes.txt on myCourses) that range from the awe inspiring to the nonsensical. Your assignment is to build a basic information retrieval system using a binary search tree containing all of the unique terms (words) in the quotes collection and to perform queries on the terms in the tree.

To build your information retrieval system, you must perform several steps:

1. Read the quotes into memory.
2. Tokenize the quotes into individual terms.
3. Convert all terms to lowercase.
4. Count how many times each individual term has occurred.
5. Place all terms, along with their counts, into a binary search tree such that an inorder traversal will display the terms in alphabetical order, along with their respective counts.

Once the binary search tree has been built, you will be implementing three types of queries.

1. Single term query - either the term is in the quotes collection or not.
2. Two term conjunctive (AND) query – this query is successful if both Term 1 and Term 2 are in the quotes collection.
3. Two term disjunctive (OR) query – this query is successful if either Term 1 or Term 2 or both, are in the quotes collection.

Requirements/Specifications:

When you construct your binary search tree, you'll be inserting objects of a class that you'll be creating called TermEntry.

TermEntry shall contain a String attribute called term and an int attribute called count, along with get methods for each attribute. Set methods are not needed.

All code should be contained in a single file called IRSystem.java. You may choose to either use a constructor or static methods in the IRSystem class. The TermEntry class and the Node class for your binary search tree should be included in this file as outer classes. In addition to reusing appropriate methods from the Binary Search Tree Exercise, you'll be implementing the following methods in the IRSystem class.

- String[] parse(String filename) – This method will read the quotes collection into memory, perform tokenization, and convert the terms to lowercase. It returns a String array containing all of the terms in the collection. Note that at this point, many terms will appear multiple times in the String array.

- void countTerms(String[] terms) – Counts how many times each individual term appears in the quotes collection. **Hint**: the best way to do this is to use another data structure that we talked about in class. Make this data structure a class member variable (or static) so that it can be seen by all methods in the class.

- TermEntry singleTermQuery(String term) – Searches the binary search tree for a given term. If found, it will return the TermEntry object containing that term. Otherwise, it will return null.

- boolean andQuery(String term1, String term2) – Performs a conjunctive (AND) query on the binary search tree for the two terms. If both appear in the tree, return true, otherwise, return false.

- boolean orQuery(String term1, String term2) – Performs a disjunctive (OR) query on the binary search tree for the two terms. If either one or both terms appear in the tree, return true, otherwise, return false.

Your code and a PDF document containing screenshots with your Test Case output is due in the Assignment 2 dropbox by the due date specified on myCourses.

Test Cases

1. After building the binary search tree, execute an inorder traversal, which outputs each term **and** the term count for each term. For example, when displaying the term “all”, the output would be “all: 4”, since that term appears 4 times in the quotes collection.

2. Perform a single term query for the term “all”. Output whether or not the term was found and if it was found, what its term count was.

3. Perform a single term query for the term “carrying”. Output whether or not the term was found and if it was found, what its term count was.

4. Perform a single term query for the term “robot”. Output whether or not the term was found and if it was found, what its term count was.

5. Perform a conjunctive (AND) query for the terms “seattle” and “mariners”. Output whether or not the query was successful.

6. Perform a conjunctive (AND) query for the terms “seattle” and “pilots”. Output whether or not the query was successful.

7. Perform a disjunctive (OR) query for the terms “four” and “score”. Output whether or not the query was successful.

8. Perform a disjunctive (OR) query for the terms “five” and “score”. Output whether or not the query was successful.

9. Perform a disjunctive (OR) query for the terms “five” and “robots”. Output whether or not the query was successful.

R·I·T	Rochester Institute of Technology Golisano College of Computing and Information Sciences School of Information

ISTE-222 Assignment 2: Non-Linear Data Structures Grading Sheet Name:

Item	Possible points	Earned points
Functionality: (80%)		
Parsing (10%) The quotes collection is read into memory All terms are converted to lowercase All quotes are tokenized and stored in a String array	 3 2 5	
Term Counting (30%) Each unique term is stored in a data structure to assist in term counting Term counts are correct for each unique term	 20 10	
Tree Building (30%) Binary search tree contains correctly implemented TermEntry objects Binary search tree is balanced and inorder traversal shows terms in the correct order and outputs their respective counts	 10 20	
Queries (10%) Single term query returns a correct result as specified Conjunctive (AND) query returns a correct result as specified Disjunctive (OR) query returns a correct result as specified	 2 4 4	
Style: (5%) Class members are private if a class is used or static if static methods used	 5	
Efficiency: (15%) The data structure used to help calculate the term counts is chosen based on its efficiency with regards to speed of operations	 15	
Total:	100	

Comments: