COP 3530

Programming Assignment #4

A **Trie** (pronounced "*try*") is a type of search tree that is used to represent key-value relationships efficiently in cases where the key is a string of characters.

It has the advantage of being able to associatively map data to string keys with search and insert complexities unaffected by the number of keys in the data structure, much like a Hash Table. This makes them a good option for implementing things like dictionaries.

See https://en.wikipedia.org/wiki/Trie for a more thorough description an an example of a Trie.

Assignment Requirements

For this assignment you must Implement a Trie that will use String keys to store integer counts of each word in a sentence. Your Trie must...

1. Support the methods `increment(char[] key)` and `decrement(char[] key)`. These methods will work as follows:
    - If the Trie already contains a leaf node representing the argument String:
        - `increment()`: Increment the count associated with it.
        - `decrement()`: Decrement the count associated with it.
    - If the Trie does not contain a leaf node representing the argument String:
        - `increment()`: Insert the nodes necessary to represent the String, and initialize the leaf node's associated count to 1.
        - `decrement()`: Do nothing.
2. Have a `postOrderTraverse()` method that will traverse every node in post-order traversal order, and if the node is a leaf, print the word that it represents, as well as its associated count, separated by a colon and a space (e.g. word: 1).

To demonstrate your Trie, you must write a main method that reads **two** String sentences, one at a time.

You will use the first sentence to create a Trie representing its word counts. For this, you will use the `increment()` method.

You will then use the words in the second string, and for each one, call `decrement()` on your Trie.

Lastly, your will output the contents of your Trie using the `postOrderTraverse()` method

Example

INPUT:

```
  the car and these cats are the best
  and the cat
```

Visual aids (Not part of your program)

```
State of the Trie after the first line:

|    -    -    -    -    Root Node    -    -    -    -    - |
|                    |              |                         t
a--------|           b              c                         |
|        r           |              |                         h
n        |           e              a------------|            |
|        e(1)        |              |            t           e(2)
d(1)                 s              r(1)         |            |
                     |                          s(1)         s
                    t(1)                                      |
                                                             e(1)

Trie after the second line:
|    -    -    -    -    Root Node    -    -    -    -    - |
|                    |              |                         t
a--------|           b              c                         |
         r           |              |                         h
         |           e              a------------|            |
        e(1)         |              |            t           e(1)
                     s              r(1)         |            |
                     |                          s(1)         s
```

```
                    t(1)                                  |
                                                        e(1)
```

OUTPUT:

```
are: 1
best: 1
car: 1
cats: 1
these: 1
the: 1
```

NOTES

- When decrementing the count for a key, if the count reaches 0, the node is marked as no longer being a leaf, meaning that it no longer represents a word. If it has no children, it is deleted from the Trie.
- `increment(char[] key)` and `decrement(char[] key)` must run in **O(m)** time, where m is equal to the number of characters in `key`.
- For the purposes of post-order traversal, if a node has multiple children, they are traversed in alphabetical order.
- You are only responsible for dealing with input in the form of two lines of space-separated lowercase English words. You are not responsible for dealing with any punctuation or input validation.