ID1020 – Lab 4
Tree

1 Organisation

This lab follows the same schema as Lab 2. It consists of

a) a preparation test in the form of multiple choice questions (marked with MCQ) and

b) a programming part that is to be *presented orally* in the lab session (redovisning) (marked with a *C*).

All the questions must be answered in Canvas. For each programming task, you need to upload your java files into its designated answer box in Canvas.

2 Goals

This lab has the following goals:

- Reason about search trees
- Work with different tree-like structures
- Work with different forms of graph traversal/search

3 Tasks

3.1 Preparation Test (MCQ) – 40P

Look at the lab assignment in Canvas.

3.2 Counting Trie (C) – 40P

A *Trie* or *Prefix Tree* is an associative array tree data structure where the *key*s are encoded in the structure of the tree (as opposed to being stored at a specific node) and any node can have a *value* which would then correspond to the key encoded in the path from that node to the root. As example consider figure 1 which encodes the following key-value-pairs: (”a”, 0), (”ab”, 1), (”ac”, 2), (”b”, 1), (”ba”, 0), (”bad”, 1), (”baf”, 3)

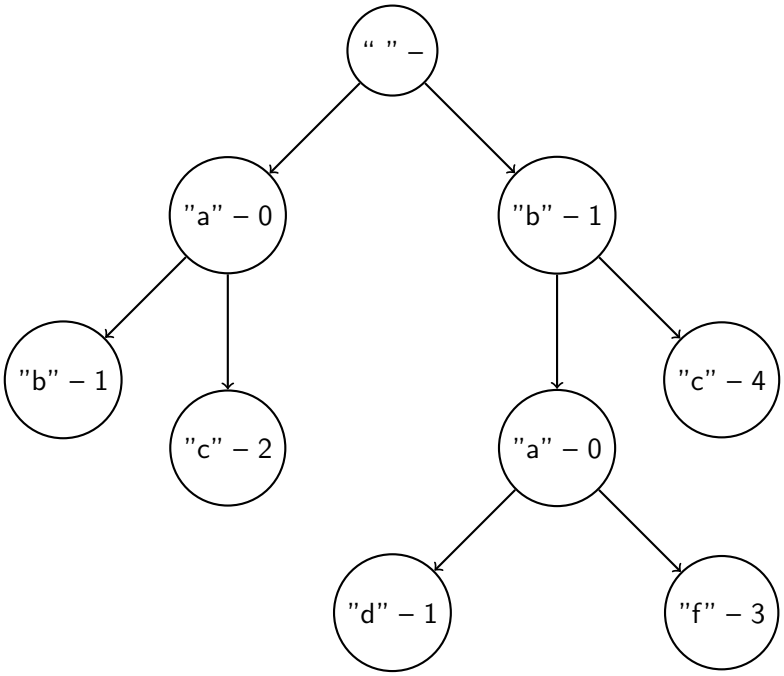

Figure 1: A Trie

and (”bc”, 4).

Your task is to implement a Trie for Strings in Java. It should support the following operations:

Put a key k — inserts k into the structure. If k was not associated with any value v before, associate it with 1, otherwise with $v + 1$ where v was the old value.

Get a key k — return the associated value for k or 0 if no value is associated.

Count on a prefix k — return the sum of all associated values of the sub-tree starting at k. For example Count(”ba”) on figure 1 should return $0 + 1 + 3 = 4$.

Distinct on a prefix k — return the number of distinct keys that have associated values on the sub-tree starting at k. For example Distinct(”ba”) on figure 1 should return 2 (for the keys ”bad” and ”baf”).

Iterator on a prefix k — return an implementation of java.util.Iterator<java.util.Map.Entry <String, Integer>> such that k is a prefix for all keys in the entries of the iterator and the keys are returned in ascending alphabetical order. For full points do not use an intermediate representation (e.g. a list) but have the iterator traverse the underlying tree structure directly (-5P otherwise).

Note: key k in all above queries is a String such as "a" or "cat".

3.3 Hot Words (C) – 20P

Write a Driver class that uses the book "Tale of Two Cities" at http://www.gutenberg.org/cache/epub/98/pg98.txt to generate the keys for your trie from section 3.2. For simplicity split the input on spaces and assume the result is a list of words. Then add all the words into the trie. Use the methods defined in section 3.2 or expand your code to answer the following questions in Canvas:

1) What are the 10 words with the highest frequency, and what are their corresponding frequency?

2) What are the 10 words with the lowest frequency, and what are their corresponding frequency?

3) Which prefix of length 2 has the highest frequency?

4) What is the letter that the most different words start with? (Not frequency this time.)

Note: If you place the text from the book in a file <maven root>/src/main/resources/kap1.txt you can load it with the Driver code in Canvas. The code also does some pre-processing of the input, but as you will see it is far from complete. Written text has many edge cases when it comes to usage of special characters and dealing with all of them would take many more lines of code than seems appropriate for this task. You are welcome to experiment with different filters, but remember that it will change the results of the questions above.