a Bag of Make Me Words

January 7, 2017

A standard part of text analysis—for machine learning, machine translation, sentiment analysis, and so on—is transforming a document into a "bag of words" representation. A "bag of words" is a data structure that maps the unique words in the document to the number of times each one appears. For example, the phrase "make me a hot dog, a chili dog, and a bag of words" would become a 'bag' like: [a:3, and:1, bag:1, chili:1, dog:2, hot:1, make:1, me:1, of:1, words:1], with each word and its number of appearances. (We put the words in sorted order, but that's not necessary.) Note that the phrase has 13 words total but only 10 unique words (because "a" is repeated twice and "dog" once in the original phrase).

Specifically, you want to create an algorithm that—given a list of words W containing m words total but only n unique words, where $n \leq m$—produces a complete list of tuples (pairs) of words and their numbers of occurrences. The result should have n entries (exactly one for each unique word) and the total of the numbers of ocurrences across all entries should be m, but these entries can be in any order.

Most approaches we consider will use hash tables. For our purposes, such a hash table supports 6 operations. Here they are described for a standard hash table using chaining:

ConstructHashTable(e) creates and returns a hash table of size e. (The hash table will be empty, but its underlying array will have e entries. Takes time proportional to e.)

Size(T) returns the number of keys in hash table T. (Takes constant time.)

Contains(T, k) returns true if hash table T contains an entry for the key k and false otherwise. (Takes expected constant time, worst-case linear time.)

Insert(T, k, v) inserts key k with value v into hash table T. If k is already in T, overwrites its value with v. (Takes expected constant time, worst-case linear time.)

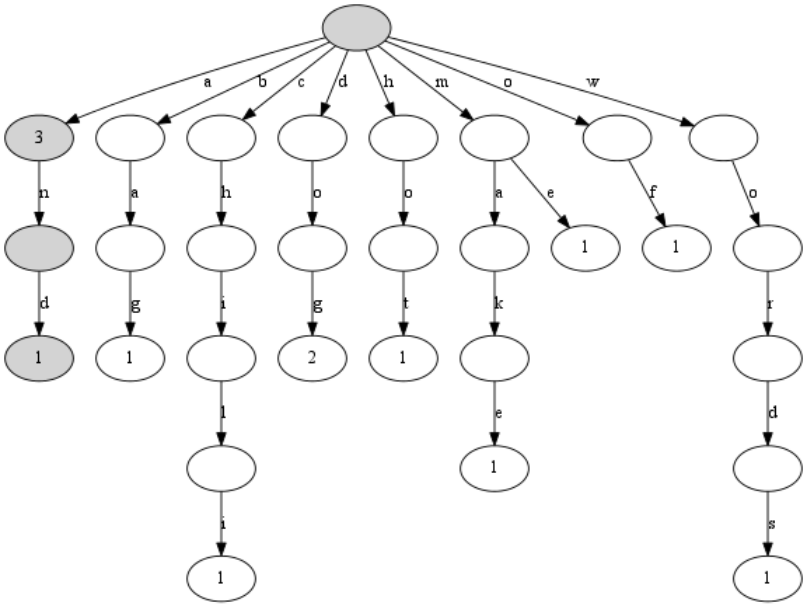
Find(T, k) finds key k in hash table T and returns the value associated with it. If k is not in T, produces an error instead. (Takes expected constant time, worst-case linear time.)

Entries(T) returns a list of all the key/value pairs in hash table T. (If the table currently has n keys (i.e., Size(T) $= n$) and its underlying array has e entries, this takes time proportional to $n + e$.)

1 Some Do and Others Trie

A *trie* is a tree data structure for storing key/value pairs where a key can be represented as a word (a string of letters). The trie has a fixed alphabet, like the 26 letters of the English alphabet. The root node of the trie represents the empty string. Each node stores a list of child pointers, one for each letter in the alphabet. Each node that represents a complete word stores the value associated with that word. (Practically speaking, nodes that don't represent complete words store some kind of "null" for their value indicating that no complete word ends at that node.)

For example, here is a trie that represents the bag of words [a:3, and:1, bag:1, chili:1, dog:2, hot:1, make:1, me:1, of:1, words:1]:



We've shaded the leftmost path in this trie. The first shaded node under the root represents the word "a", which appears 3 times. The bottom node along that shaded path represents the word "and" (because we follow pointers from the root labelled "a", "n", and then "d" to reach it), which appears 1 time. The word "an" wasn't in our bag, and thus the second node along the path has no value.

CONTINUED ON THE NEXT PAGE

1. Alter the sketch above to add the following to the trie:

   - add the key "do" with value 4
   - add the key "chime" with value 2

2. We described an upper-bound on the worst-case runtime of operations on the hash tables in terms of the number of keys stored in the table. That's not the most convenient variable to describe the runtime of operations on the trie, however. In terms of what quantity (variable) do operations on the trie run in worst-case linear time?

3. **[WORTH 1 BONUS POINT ON THE ASSIGNMENT AND COURSE; NOT REQUIRED OR GRADED FOR THE QUIZ.]** We **can**, however, give a lower-bound (an Ω bound) on the worst-case runtime of operations on a trie in terms of **just** the number of keys stored in the trie. Give and briefly explain the bound.