COSC 320 - Advanced Data Structures and Algorithm Analysis Project 2

Dr. Joe Anderson

Due: 10 November 2019

1 Description

For this project, you will implement an English-language spell-checking program that uses hash tables. The program will read a dictionary file (provided on the course webpage) and store it into a hash table. Then it will let a user type some text (anywhere from a single word to multiple paragraphs) and report any misspelled words, and possible corrections.

For information on the theory behind hash tables, see the textbook and lecture notes.

2 Specifications

1. Create a class called `Dictionary` that will be a hash table structure, containing an array of pointers, to a chain (doubly-linked list) of values and keys that each map to that cell of the array under the chosen hash function.
2. You may use either the division or multiplication hash functions discussed in class, with the standard interpretation of strings into natural numbers. For either method, make sure that it takes advantage of the size of your hash table (at least 80% of the cells should have at least one element).
3. Use the file `english.txt` provided on the course webpage as the starting data for the has table; when the program starts, it should read the file and insert every word in to the hash table.
   (a) After the dictionary is loaded, print some statistics about the hash table: total words in the table, largest bucket size, smallest (non-zero) bucket size, how many total buckets there are, how many buckets are used, what is the average number of items in each bucket, and how long it took to load and insert them all.
4. After the user enters some text, the program should search the dictionary for each string.
   (a) If it is found, then it is assumed to be spelled correctly
   (b) If the string is not in the table, then the program should find all valid string withing 1 or 2 edit-distance of the candidate string
   (c) The "edit distance" is defined as number of the following operations needed to obtain one string from the other:
      i. Replacing one character
      ii. Adding one character
      iii. Deleting one character
      iv. Swapping two adjacent characters

(d) There will be *many* possible strings obtained by 1 or 2 edits, but they should be filtered by which are actually valid words in the dictionary. For example: the following are just some of the strings within one edit distance of `word`: `wird`, `wrd`, `work`, `sword`.

(e) If there are no suggestions found, e.g. if the user enters `aaaaaaaaaaa`, print a message to denote that no corrections are available.

(f) Report the following statistics about the text: how many misspelled words, number of possible corrections found, and the time it took to compute the suggestions.

(g) Avoid including punctuation in words. Contractions, numbers, sentence structure symbols, etc. should mostly be permitted.

5. Include a `Makefile` to build your project.

6. Include a README to document your approach to the problem, any interesting design choices you made, any deficiencies you found, and improvements you would make. Document how you interpret the strings as keys, and give some detail about your choice of hash function.

3 Submission

Submit a `.zip` file called `Project2[LastName].zip` (with `[LastName]` replaced with your own last name) containing your source code, `Makefile`, and documentation, then upload it to the course MyClasses submission page.

Print out your code and documentation, to be turned in during the first lecture following the due date.

4 Bonus

(10 pts) Prompt the user to accept each of the suggested replacements, then print the corrected text back to the user. Allow the user to opt-out of the suggested replacements and save their word into the dictionary file for later use (e.g. they may want to save proper nouns that wouldn't otherwise be accepted).

(10 pts) Implement multiple hash functions (multplication and division), and do a (thorough) performance comparison of each.

(5 pts) Use the terminal color characters (Google will be your friend here) to highlight the misspelled words in the text

A Example program output

```
[jtanderson@linuxlab spellcheck]$ ./spellcheck
Welcome to Spell Checker!
---------------------------------------------------------------
Loading the database...
---------------------------------------------------------------

Total words = 466544
Biggest bucket size = 971
Smallest bucket size = 1
Total number of buckets = 10000
Number of used buckets = 9620
Average number of nodes in each bucket = 46
Total time taken = 0.360616

---------------------------------------------------------------
```

```
Please enter some text:
-----------------------------------------------------------

This is a test. Thisi is another test.

Thisi is misspelled! Below are the words within one edit distance
-----------------------------------------------------------

Suggestions for thisi: *thiasi* *thitsi* *thisn* *this*


The following are within two edit distances for all words
-----------------------------------------------------------

Suggestions for thiasi: *thiatsi* *thiazi* *thiasoi*

Suggestions for thisn: *shisn* *thins* *hisn* *thin*

Suggestions for this: *chis* *ghis* *khis* *phis* *tmis* *tris* *tuis* *thais* *theis*
*thos* *thus* *thia* *thig* *thio* *thir* *his* *tis* *thi*


-----------------------------------------------------------
Summary
-----------------------------------------------------------
Number of misspelled words = 1
Number of suggestions 29
Time required to find suggestions = 0.000950549
```