Nathan Chen

UID: 604902470

Project 4 Report

In the depicted diagram of my Project 4 solution, we use a hash table formed from a fixed array, with each array item containing a Node pointer that was part of a linked list of Node pointers. Each Node had member variables that indicated the value/word or phrase, the key that represents the bucket position, and the super key with a unique ID for all anagrams, along with a pointer to the next node (nullptr if there is no next node).

Here, characters are mapped to certain keys/super keys, and prime numbers are used to do this to ensure unique IDs for the super keys of different words, while maintaining the same ID for anagrams of the same word. Sometimes an anagram that doesn’t match may land in the wrong bucket, so we have the superkeys in order to make sure that we avoid this; the non-matching anagram doesn’t have a matching superkey, so it is distinguishable.

nullptr

nullptr….

……….

29

17

euphemism

next

next

5

3

god

5

3

dog

null

null

**Pseudocode:**

int hashFunc(word)

{

set “result” (i.e. return value) to -1

clean up text, leave only necessary letters and put in all lowercase

initialize the super key value “primeMult” to 1

initialize array “prime” with a capacity “LIM” to have the 26 first prime numbers

for all the characters in the provided text

{

each letter’s corresponding prime number in the prime array is multiplied by the other letters’ prime numbers, and then the super key is set to the product

}

set result to modulo of superkey divided by hash table size (aka bucket number)

return result

}

void insert(word)

{

execute hashFunction(word) and set the return value to hashkey “key”

initialize a new node pointer to point to index of hash table with hash key

if the value is a null pointer

{

assign a new node in that position, with parameters “word” and “key”

}

else

{

initialize a new temporary node with parameters “word” and “key”

initialize a node pointer and point it to the start of the bucket

while (pointer isn’t a nullptr)

{

increment pointer forward once

}

previous node pointer points to the temporary node

}

}

void lookup(word, callback function)

{

if the callback function returns a nullptr

{

return

}

if the phrase passed to the parameter is empty

{

return

}

call hashFunction(word) and set the return value to hash key “key”

get super key with callSuper

initialize a pointer “xx” to point to the key index of the bucket

while (“xx” isn’t a null pointer)

{

if (xx’s super key is == callSuper)

{

execute “callback” function on pointer

move pointer forward by one

}

else

{

move pointer forward by one

}

}

}

No known bugs, serious inefficiencies, or notable problems