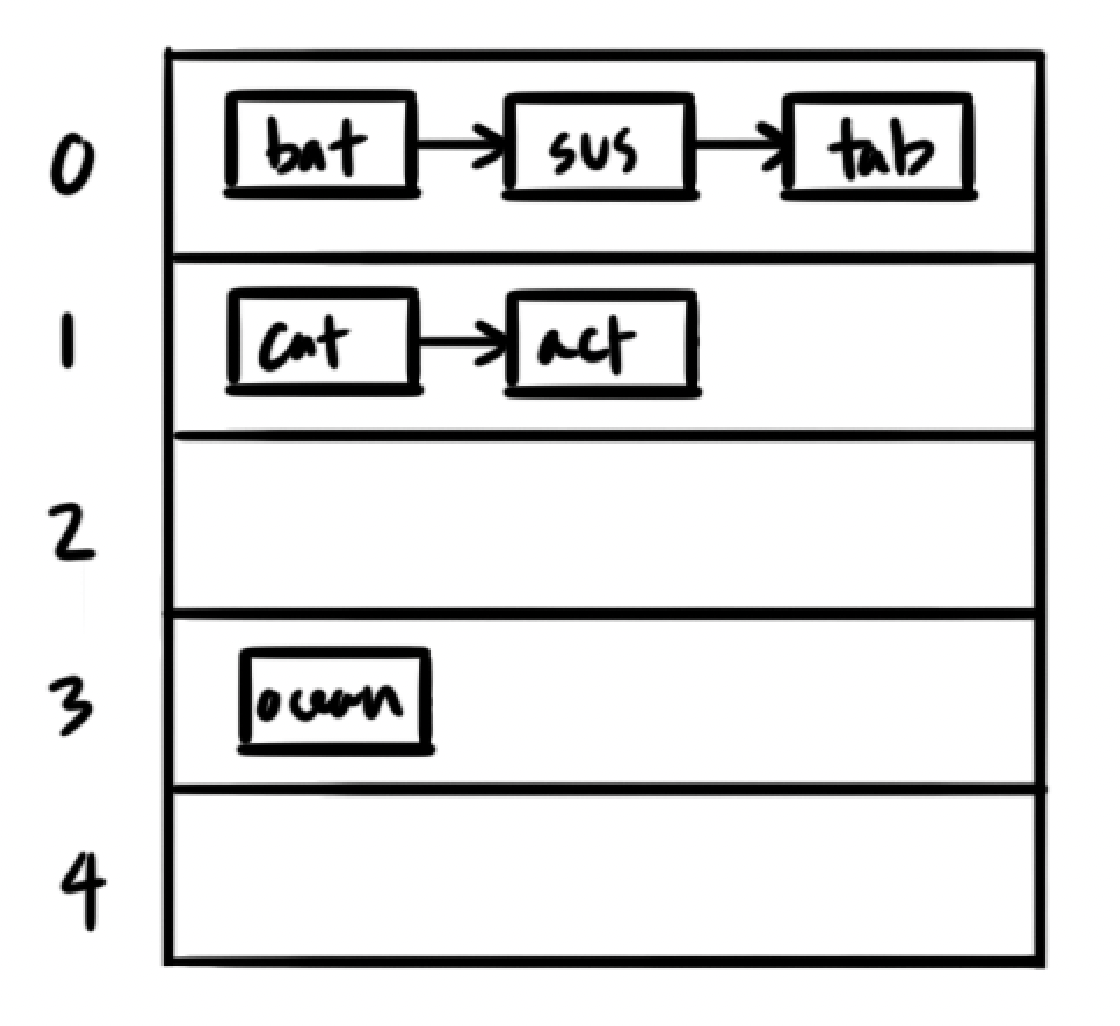
Report

**Description**

DictionaryImpl uses a hash table to store and lookup words. The constructor initializes m\_hashTable, which is a vector of lists of strings (vector<list<string>> m\_hashTable), with a size of maxBuckets. Aside from the functions already provided (such as the destructor, insert, lookup, and removeNonLetters), there is also an additional hashFunction function that takes a word as a parameter and returns the hash value of the word. The generateNextPermutation function was removed since using a hash table makes it unnecessary to generate permutations.

Because DictionaryImpl utilizes a hash table and the hash function assigns a word a hash value based on the sorted version of the word, all anagrams have the same hash value and are therefore assigned to the same list/index in the vector. To lookup the anagrams, you find the index in the vector the original word belongs in and print everything in the list as long as the word you’re looking up is the same as the word in the list when both are sorted.

**Psuedocode**

**int DictionaryImpl::hashFunction(string& word) const**

sort word

use the hash function on the sorted word, cast it to an int, and return it

**void DictionaryImpl::insert(string word)**

format word

if word is not empty

make a copy of word

call the hash function on the copy and % by the hash table size

insert word in the hash table

**void DictionaryImpl::lookup(string letters, void callback(string)) const**

if callback is a nullptr

return

format letters

if letters is empty

return

call the hash function on letters and % by the hash table size (letters gets sorted)

retrieve the linked list at the bucket index

for every word in the list

create a copy of the word

sort the copy

if letters is the same as the sorted version of the word

call the callback function on the word

**Problems**

As of this moment, there are no known bugs or serious inefficiencies. The algorithm may be able to run faster without the usage of the function removeNonLetters or with a more efficient version of that function. However, because this function was given, I did not want to needlessly change it. A notable problem while working on this project was figuring out how to implement the hash table. But once I was able to figure that out, it was a relatively smooth process coding everything.