Tianyi Li

CS 1501  
Dr.John Aromis

Project 2

An anagram is a word game. It used a word or phrase to form by trying all possible combination. A unorder phrase that can break to with few different words. That a secret message can be deciphered.

The Anagram Problem is a famous hash table problem. A description of the problem is as follows:

Give a secret un order message, it may can form to 1 or 2 or more word, for each different started of words that can to form is same or different way. If there a same word that form is should not be store in the result value. Sometime can not find word should to skip. What is the best way to decoder this secret message?

In the project, the original message can be store in the string. Also, the program needs to pass in the dictionary file. The dictionary file can be read by file reader and buffered reader. After read in the dictionary file, insert the dictionary file two time. Fist insert in to a hash table called table(dic), a hash table one by one as prefix table, the prefix table is mainly to check all the start char that can be forming to word. Then I make contain, the contain have 2 part. First, check if the word is in the dictionary (hashtable table). The second is to check if the will form word in the prefix or not. The follow put can be showing in follow pseudo code:

Do insert( word ){

Put all word in hash table dic(table);

For (n<word length){

Put each char at the n is the hash table prefix;

}

}

Containdic( word ){

Word is containing in dic(table)

}

Containprefix( word ){

Word is containing in prefix

}

There is one part pass the different letter of word in the program by using the for loop to loop trough the word, and called recursive method to get the result.

Convert the word to a char of array

For ( word length){

Remove word

Recursive method(the start of word, the array of char that left, “”(result))

Add the word back on.

}

The last recursive method to get result.

If(the passing in start in the prefix or the start is empty){

To remove next char

Do recursive

Add the word back again

}

If(start is a word ){

Save the original result + result + “ ”

If (the array of char is empty and the result is not saved before ){

Print it out;

}else

Do the recursive again because is not a word

}

I think my efficiently is good but can be improve, the hash table is fast by only check the key contain. By using the checking the prefix can be same a lot of time by without checking the all the full string. The only thing that I thick can improve the runtime is to use the swap method that can swap the order of the word. For example: Let it go can be form to it let go, it go let, let go it, go let it, go it let. If those answer can be store that can same runtime.