CS380 Algorithm Design & Analysis

Assignment 4: Anagram Game

Date Assigned: Tuesday, March 31, 2009
Date Due: Thursday, April 9, 2009
Total Points: 45pts

For this assignment, you are to implement an anagram game. The game will present a word to the user and the user must try and come up with as many anagrams for that word as possible.

For example, if the program produces the word “tar”, the user could guess the word “art” and “rat”. The user can play as many times as they like and you are to keep a running score of how they did. For every correct word that they produce you should add 5 points to their score, and for every incorrect word you should subtract 5 points. The user should be given the option of ending a round at any time, and ending the game after ending any round.

For the purpose of this assignment, we are only interested in words that appear in the dictionary. The dictionary that we are using is the Scrabble TWL dictionary of four letter words. It is freely available from http://members.ozemail.com.au/~rjackman/4fours.html and I have also placed it on the CS380 public folder on Turing.

Since we will be performing multiple anagram queries, our first step is to load all of the words in the TWL into an appropriate data structure. A primary requirement is that one must be able to efficiently search this data structure to look for anagrams of a given word.

A clever trick that we will use to facilitate this is to work with *alphagrams* of words, not just the words themselves. The *alphagram* of a word (or any group of letters) consists of those letters arranged in alphabetical order, and is the way expert Scrabble players sort the tiles on their rack. So, the *alphagram* for the string “toxic" is “ciotx", similarly the *alphagram* for both “star" and “rats" is “arst".

Now what we will do is store words and their *alphagrams* into a hash table. For each word, we will compute its *alphagram*. We will treat the *alphagram* as the key, and the word as the value of a <key, value> pair. Each pair is then inserted into a hash table, where the hash is applied on the key, but the entire pair is stored. This approach guarantees that all words which are anagrams of one another are stored in the same bucket of the hash table, since those words have common *alphagrams*. It will be helpful to define a new Word class to store the pairs and provide any additional Word functionality.

You should feel free to use the methods described in class and in the text for appropriate hash functions for hashing strings (but please cite any source which you use).

What to Submit

- Submit an electronic copy of your project by 9:15am on the day that it is due. Name your project “04PUNETAnagram”, replacing PUNET with your PU Net ID (i.e. khoj0332).
- A summary of the time that you spent working on this assignment, and what slowed you down the most. Submit this document electronically as a Google Document called “04PUNetAnagram” for example “04Anagram”. Create the Google document and share it with me at ShereenKhoja@gmail.com.