24-780 B—Engineering Computation

Assigned: Wed. Nov. 15, 2023
Due: Tues. Nov. 21, 2023, 11:59pm

Problem Set 9: Word Count

In past semester, I have asked students to implement a binary search tree (BST) and use it to count the frequency of words in a specific text. For PS09, we will follow on that idea but, instead of implementing our own BST, we will make use of the data structure available in the C++ library, namely std::map.

Task 1 (Generate Word Count)

To start, create a class called *WordCollection* that can store all the words (strings) that are asked to be included. When a word already exists in the collection, the count for the word is increased. This means that the storage variable of type std::map must use std::string as a key, and int as the storage item. The main() function and the readFile() functions for using your newly created class are provided to you (try not to change them too much). Here is a list of the needed functions:

```
// Insert item into Collection (BST). If newItem is invalid, do nothing
// If item in BST, update the item's count, o/w insert it and make count = 1
virtual void insert(const std::string &newItem);
void operator+=(const std::string& newItem);

// Remove item from Collection (BST).
// Reduce count if item is in collection. Eliminate item if its counts goes to zero
virtual bool remove(const std::string &wantedItem);
void operator-=(const std::string& wantedItem);
virtual bool removeAll(const std::string& wantedItem);

// Find the item and return the item count.
// Returns zero if the item is not in the collection
int itemCount(const std::string& wantedItem);

// Traverses the collection and prints out to a stringstream each of the items
// in the BST in correct order, along with the item's count, separated by '\n'
void printInOrder(std::stringstream& out);
void printInOrder(std::ostream& out);

// Returns number of unique words
int uniqueWordCount();

// Returns total of all the words
long int totalWordCount();
```

Task 2 (Word Cloud)

I used the word frequency developed in *WordCollection* class to create a ***very, very simple*** word cloud, making use of the *GraphicFont* library I developed (which is included in the zipped project file).

Note that the word cloud used in the title above requires that the placement of each word be perfectly located based on pixel reasoning so that the cloud can fill in tiny spaces between words. I did not accomplish this level of sophistication for this assignment. Instead, I created a word cloud using the following simple rules:

- User can select how to display rare words
- Some common words (e.g., “the”, “a”, “and”, etc.) can also be omitted using a pre-loaded list

- Whatever size (*S)* is selected for words with a count of 1, the size is increased by a factor for each increase (e.g., 2*S*, 3*S*, 4*S*, 5*S*, 5*S*, and 6*S* for word counts of 2, 3, 4, 5, and 6)
- The location of each word is randomly selected for each word (i.e., random X, random Y).
- The transparency of each word is set to about 0.6 so that you can see words that are covered up.
- The colors can be randomized (i.e., rainbow)

Feel free to play around with the graphics so you get to understand how it all works.

Task 3 (Make it faster)

In Task 1, we implemented the *WordCollection* class using a Binary Search Tree (std::map). Switch the implementation to a Hash Table (std::unordered_map) and compare the time performance for the following 2 cases (determine the time difference using each of the implementations):

a) Reading the CountOfMonteCristo.txt file (This part is already in the main() function)

b) Again using the CountOfMonteCristo file, removing and then re-adding the word "count" 1 million times (you need to develop necessary code)

Provide your very brief yet numerically meaningful answers in a comment section at the end of the WordCollection.h file

Deliverables

1 zip file, containing your whole project. I advise that you include your name in comments at the top of each of the files. I expect your project will at least include the following:

wordCloud_main_andrewID.cpp << contains *main()* and some of your code

WordCollection.h, WordCollection.cpp << contains your new code

Upload the zip file to the class Canvas page before the deadline (Tuesday, Nov.21, 11:59pm), although I expect that you'll finish this assignment rather quickly so you can work on the Team Project.

Learning Objectives

Template classes in C++

Inheritance in C++

Using Binary Search Trees (std::map)

Using Hash Tables (std::unordered_map)

Use OpenGL primitives along with code written by others.

Searching references (online or textbook) for C++ library functions.