Assignment 9 and 10

Data Structures in C++ and Python

EC602 Fall 2016

Contents

1 Introduction

Storing, managing, searching and understanding information is a critical task that all electrical and computer engineers must deal with and be prepared for.

One critical component of this is the use of appropriate data structures and algorithms.

This week, we challenge you to select among many possible data structures and algorithms to solve a problem.

1.1 Assignment Goals

The goals for assignment 9 are to

- design an efficient algorithm using python data structures and standard library modules
- write Python code that passes a static code-style analyzer.
- write Python code that is also elegant.
- explore algorithms in pursuit of solving assignment 10.

The goals for assignment 10 are to

- design an efficient algorithm using STL data structures.
- write C++ code that passes a static code-style analyzer.
- write C++ code that is also elegant.

1.2 Due Date

This assignment is due 2016-11-21 at midnight.

1.3 Submission Link

You can submit here: week 9 submit link and week 10 submit link

2 Data structures

2.1 Operations on a Data Structure

We consider dynamic data structures The key operations are:

- insert: inserting a new element
- search: finding an element
- sort: retreiving all the elements in ascending or descending order
- delete: deleting an element

2.2 Types of Data structures

There are three very important types of data structures we will consider

- hash tables (python dictionaries). A hash table has $O(1)$ insert and $O(1)$ search. However, the data is unsorted, so sorting would still require $O(N \log N)$.
- binary search trees. A binary search tree has $O(\log N)$ insert and $O(\log N)$ search.
- heaps or priority queue.

3 Containers in Python

Python does not have a built-in binary search tree or one that is part of the standard library.

There is speculation around why this is the case.

One theory is that there are so many flavors of BST that it would have been impossible to pick "the right one" to include in the standard library.

Another theory is that the BDFL GvR thinks that you should almost always use lists or dictionaries instead of a BST.

4 Containers in C++

The following STL containers are variations on binary search trees:

- `map`
- `multimap`
- `set`
- `multiset`

Maps contain *elements* in addition to their *key*.

The following STL containers are variations on hash tables / dictionaries:

- `unordered_map`
- `unordered_multimap`
- `unordered_set`
- `unordered_multiset`

The following STL container implements a priority queue or heap:

- `priority_queue`

In the above, the word `multi` indicates that there can be multiple values of a particular `key` in the structure.

5 Style

5.1 Python PEP8

Python style is governed by an official document PEP8 and accompanying software `pep8`. The software is available both as a module and as a command-line program.

You can test your programs for compliance using

```
pep8 my_program.py
```

5.2 C++ lint

The python script cpplint is available from github, and can be installed from the command line as follows:

```
pip install cpplint
```

cpplint assesses your C++ programs according to google's C++ style guide

You can test your C++ programs for compliance. In a unix/linux terminal, type

```
cpplint my_program.cpp
```

6 The assignment: wordplayer

Write a program to read all the words from a word list and then use this to answer what words can be formed from letters.

The wordlist is specified using a required command-line argument.

You must write both a python version and a C++ version.

Use the python version to explore the design space and test out ideas, then use the C++ version to

6.1 Program Behavior

The filename of the wordlist is provided on the command line. It will contain all the defined words, one per line. Each word will only appear once.

In this problem, all words will be lower case.

After reading and "digesting" the information in the wordlist, the program should interact with the user using the terminal input and output.

The user will enter letters followed by a number N, and the program will respond with all the N-letter words that can be formed using those letters.

The words should be printed in alphabetical order, one per line, followed by `.` on its own line. This helps make the output more readable when multiple queries are made and the result is stored in a file.

If number is 0, the program exits.

Here is an example:

```
% w9_wordplayer big_wordlist.txt
testthisprogram 11
prostatites
prosthetist
seismograph
straightest
thermistors
thermostats
.
ndarmo 5
adorn
manor
monad
nomad
radon
roman
.
omrandn 6
random
rodman
.
exit 0
%
```

The lines with numbers represent user input, the other lines are the answers.

So, when you type

```
testthisprogram 11
```

the program responds with

```
prostatites
prosthetist
seismograph
straightest
thermistors
thermostats
.
```

which are the 11-letter words that can be formed by selecting 11 letters from "testthisprogram".

The program continues answering problems until the number 0 is entered.

No error checking is required.

6.2 Grading Rubric

Out of 5, your grade on this assignment will be assessed as follows:

- 2 points: your code passes the unittester for specifications
- 1 point: your code passes the unittester for style (as determined by cpplint or pep8).
- 1 point: your code will be evaluated for elegance, as determined by code length. Code length is defined as the number of words. Shorter programs will score higher.
- 1 point: your code will be evaluated for efficiency, as determined by the time to process `wordplayer_testinput.txt`. Faster programs will score higher.

Your score for elegance and efficiency will be 0 unless you pass both the specification test and the style test.

You can test your program using the included files `wordplayer_input.txt` which is a 1000 line long file of letter combinations and word size targets.

The correct output for this input is in `wordplayer_results.txt`

In the terminal, you can type

```
w9_wordplayer big_wordlist.txt < wordplayer_input.txt  > mywordplayer_results.txt
```

and then compare the output you created `mywordplayer_results.txt` with my output.

My C++ program takes 0.22 seconds to run on the server.

7 Links

Here are the relevant files for you to download:

- w9_wordplayer_tester.py
- big_wordlist.txt
- wordplayer_testinput.txt
- wordplayer_results.txt