Assignment 8: Data Structures in C++ and Python: wordplayer

EC602 Design by Software

Fall 2018

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 this python part 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 the C++ part

The goals for the C++ part 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 Group Size

For this assignment, the maximum group size is 3.

1.3 Due Date

This assignment is due 2018-12-03 at 23:59:59 (one second before midnight)

1.4 Submission Link

You can submit here: `wordplayer` submit link

1.5 Points

This assignment is worth 20 points, 10 points for each of the C++ and python parts.

2 Data structures

2.1 Operations on a Data Structure

We consider dynamic data structures for storage of *dynamic sets*. 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

The following containers are part of the standard library or built-in:

- `set`,`frozenset`: `key` only; $O(1)$ lookup
- `dict`: `key` and `value`; $O(1)$ lookup
- `tuple`: immutable ordered sequence
- `list`: mutable ordered sequence

The libraries `collections` and `itertools` also have useful container types.

3.1 Binary Search Tree

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.

It is possible to make a BST using tuples of tuples or lists of lists.

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

Programs are read much more frequently than they are written, so it is important that your programs follow a consistent style to help others (including future you) be able to read your code.

Following a style guide is a useful part of writing readable code, and it is common industry practice to adopt a company-wide style guide.

5.1 Python PEP8

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

You can test your programs for compliance using

```
pycodestyle my_program.py
```

5.2 Pylint

Pylint provides a more comprehensive analysis of your code, along with a score out of 10.

You can evaluate your program using

```
pylint my_program.py
```

5.3 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
```

5.4 Autoformatters

For python, `autopep8` can be used to fix many white space formatting problems.

For C++, `astyle` does this.

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 develop the C++ version using your best ideas.

6.1 Program Behavior

The filename of the wordlist is provided on the command line. The file 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 letters are like scrabble tiles: to spell a word that has multiple letters, all those letters must be present.

For example: `peel` is an answer to `leeping 4` but it is not an answer to `plinge 4` (`plinge` only has one `e`)

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:

```
% python wordplayer.py words100k.txt
helloworld 6
holder
hollow
rolled
wooled
.
ndarmo 5
adorn
roman
.
omrandn 6
random
rodman
.
exit 0
%
```

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

So, when you type

```
helloworld 6
```

the program responds with

```
holder
hollow
rolled
wooled
.
```

which are the 6-letter words that can be formed by selecting 6 letters from `helloworld`.

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

No error checking is required.

6.2 Grading Rubric

Your grade on this assignment will be assessed as follows:

- meeting the specifications (4 points)
- style (as determined by pep8/pylint or cpplint) (2 points)
- elegance (as determined by a code complexity measurement) (2 points)
- efficiency (as determined by a code speed measurement) (2 points)

The exact scoring mechanism will be specified within the checker.

You can test your program using the included files `wordplayer_testinput.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

```
python wordplayer.py words100k.txt < wordplayer_testinput.txt  > mywordplayer_results.txt
```

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

Your python program must be called `wordplayer.py` and your C++ program must be called `wordplayer.cpp`

7 Links

Here are the relevant files for you to download:

- wordplayer_checker.py
- ec602lib.py
- words100k.txt word list with 100 thousand words
- words680.txt word list wit 680 words.
- wordplayer_testinput.txt input sequence
- wordplayer_results.txt correct output for above input.

- wordplayer_sol this is a Linux executable compiled from the reference C++ solution.

7.1 Using the Checker

A command line argument of `py` or `cpp` is required.

To check `wordplayer.py`, run

```
python wordplayer_checker.py py
```

To check `wordplayer.cpp`, run

```
python wordplayer_checker.py cpp
```

7.2 About the grades

Please note that any grades reported are tentative and are subject to review of your code for un-acceptable hand optimizations (like pre-computing the answers to the tested questions.)