CSE 332: Data Structures and Parallelism — Winter 2019

P2 Write-up

P2 Due Date: Tuesday, February 19

Only ONE person from your group should submit the write-up to Gradescope. Please make sure that you select corresponding pages for each question when you are submitting your write-up to Gradescope. You should also add your partner to your group on Gradescope after you submitted your write-up.

Project Feedback

(1) Project Experience

Answer the following questions about your experience doing the project.

- What was your favorite part of the project? Why?
- What was your least favorite part of the project? Why?
- How could the project be improved? Why?
- Did you enjoy the project? Why or why not?

Experiments

Throughout p1 and p2, you have written (or used) several distinct implementations of the Dictionary interface:

- `HashTrieMap`
- `MoveToFrontList`
- `BinarySearchTree`
- `AVLTree`
- `ChainingHashTable`

In this Write-Up, you will compare various aspects of these data structures. **This will take a significant amount of time, and you should not leave it to the last minute.** For each experiment, we expect you to:

- Explain how you constructed the inputs to make your conclusions
- Explain why your data supports your conclusions
- Explain your methodology in enough detail for us to be able to reproduce your experiment.
- Tell us what inputs you used. If generated in the code, tell us where and how. If from a file, please describe the files and include them in the experiments folder.
- Include your data either directly in the write-up or in the experiments folder
- You should include graphs of the outputs for at least a few of the questions.
- We recommend that you keep your "N" (as in "N-gram") constant throughout these experiments. (N = 2 and N = 3 are reasonable.)
- You should probably run multiple trials for each data point to help remove outliers.
- You should not need to wait for hours and hours for your program to run in order to answer a question. Do use large values for N, but not so large that you are waiting overnight for your program to run (N=1,000,000 is likely larger than you need.).

(2) BST vs. AVLTree

Construct inputs for BST and AVLTree to demonstrate that an AVL Tree is asymptotically better than a Binary Search Tree. Comparing the worst case for each structure is fine here. To do this, we expect you to show trends. You might consider fitting a curve to your results. Explain your intuition on why your results are what they are.

(3) `ChainingHashTable` (UPDATE: Above and Beyond!)

Your `ChainingHashTable` should take as an argument to its constructor the type of "chains" it uses. Determine which type of chain is (on average, not worst case) best: an `MTFList`, a `BST`, or an AVL Tree. If it doesn't make a difference, explain why. Explain your intuition on why the answer you got makes sense (or doesn't!).

(4) Hash Functions (UPDATE: Above and Beyond!)

Write a new hash function for your CircularArrayFIFOQueue (it doesn't have to be any good, but remember to include the code in your repository). Compare the runtime of your `ChainingHashTable` when the hash function is varied. How big of a difference can the hash function make (on average, not worst case)? (You should keep all other inputs (e.g., the chain type) constant.) Explain your intuition on why your results are what they are. You may find AlphabeticString useful for this experiment.

(5) General Purpose Dictionary

Compare `BST`, `AVLTree`, `ChainingHashTable`, and `HashTrieMap` on alice.txt. Is there a clear winner? Why or why not? Is the winner surprising to you?

(6) uMessage

Use uMessage to test out your implementations. Using N=3, uMessage should take less than a minute to load using your best algorithms and data structures on a reasonable machine.

- **How are the suggestions uMessage gives with the default corpus?** (here we mean spoken.corpus or irc.corpus, not eggs.txt)
- Now, switch uMessage to use a corpus of YOUR OWN text. To do this, you will need a corpus. You can use anything you like (Facebook, google talk, e-mails, etc.) We provide instructions and a script to format Facebook data correctly as we expect it will be the most common choice. If you are having problems getting data, please come to office hours and ask for help. Alternatively, you can concatenate a bunch of English papers you've written together to get a corpus of your writing. **PLEASE DO NOT INCLUDE "me.txt" IN YOUR REPOSITORY. WE DO NOT WANT YOUR PRIVATE CONVERSATIONS.**
  - Follow these instructions to get your Facebook data. **You will need to download it in JSON format.** We also recommend only selecting your messages and limiting the date range to the past year to make the process go quicker.
  - Run the ParseFBMessages program in the p2.wordsuggestor package.
  - Use the output file "me.txt" as the corpus for uMessage.
- **How are the suggestions uMessage gives wth the new corpus?**

Above and Beyond

(7) Extra Credit

Did you do any Above and Beyond? Describe exactly what you implemented.