

CS5800: Algorithms — Spring '21 — Virgil Pavlu

Homework 7 and 8

Submit via [Gradescope](#)

Name:

Collaborators:

Instructions:

- Make sure to put your name on the first page. If you are using the \LaTeX template we provided, then you can make sure it appears by filling in the `yourname` command.
- Please review the grading policy outlined in the course information page.
- You must also write down with whom you worked on the assignment. If this changes from problem to problem, then you should write down this information separately with each problem.
- Problem numbers (like Exercise 3.1-1) are corresponding to CLRS 3rd edition. While the 2nd edition has similar problems with similar numbers, the actual exercises and their solutions are different, so make sure you are using the 3rd edition.

1. (50 points)

Implement a hash for text. Given a string as input, construct a hash with words as keys, and word counts as values. Your implementation should include:

- a hash function that has good properties for text
- storage and collision management using linked lists
- operations: insert(key,value), delete(key), increase(key), find(key), list-all-keys

Output the list of words together with their counts on an output file. For this problem, you cannot use built-in-language datastructures that can index by strings (like hashtables). Use a language that easily implements linked lists, like C/C++.

You can test your code on "Alice in Wonderland" by Lewis Carroll, at [link](#)

The test file used by TA will probably be shorter.

(Extra Credit) Find a way to record not only word counts, but also the positions in text. For each word, besides the count value, build a linked list with positions in the given text. Output this list together with the count.

2. (50 points)

Implement a red-black tree, including binary-search-tree operations *sort*, *search*, *min*, *max*, *successor*, *predecessor* and specific red-black procedures *rotation*, *insert*, *delete*. The delete implementation is Extra Credit (but highly recommended).

Your code should take the input array of numbers from a file and build a red-black tree with this input by sequence of "inserts". Then interactively ask the user for an operational command like "insert x" or "sort" or "search x" etc, on each of which your code rearranges the tree and if needed produces an output. After each operation also print out the height of the tree.

You can use any mechanism to implement the tree, for example with pointers and struct objects in C++, or with arrays of indices that represent links between parent and children. You cannot use any tree built-in structure in any language.

3. (50 points)

Implement Skiplists 50 points. Study the skiplist data structure and operations. They are used for sorting values, but in a datastructure more efficient than lists or arrays, and more guaranteed than binary search trees.