CS 202 Fundamental Structures of Computer Science II
Assignment 4 – AVL Trees, 2-3 Trees, 2-3-4 Trees, Red-Black Trees

Assigned on: July 23, 2019
Due Date: July 31 2019, 23:55

1) Question Part (30 points)

Assume that we have the following balanced-searched tree implementations:

a) AVL tree
b) 2-3 tree
c) 2-3-4 tree

Starting with an empty balanced search tree, we would like to insert the following keys into the tree in the given order:

35, 55, 75, 100, 80, 85, 65, 110, 90, 95, 125

and then, delete the keys 80 110 90 (in the given order) from the tree. Note: while deleting an internal node, its inorder successor should be used as the substitute if needed.

Show the structure for the AVL, 2-3, and 2-3-4 trees after additions and deletions with sufficient detail.

2) Programming Part (70 points)

You are to write a C++ program to count the frequency (number of occurrences) of unique words in a text file. You may ignore any capitalizations and assume that words in the text file are separated with whitespaces and punctuation. For example; the word "you'll " will be counted as two different words "you" and "ll". Your program should construct a Red-Black tree and consider each word as a key.

You are to use a pointer based implementation of a Red-Black tree to store the words and their counts. Each node object is to maintain the associated unique word as a string (key), its current count as an integer, its color, and left and right child pointers. You must implement the following functions (you don't need to implement delete procedure of Red-Black tree):

- **`addWord`:** adds the specified word in the Red-Black tree if not already there; otherwise, it simply increments its count.

- **`generateTree`:** reads the input text and generates a Red-Black tree of words. In this function you should detect all of the words in the input text and add them to the tree by using the **`addWord`** function.

- **`printHeight`**: computes and prints the height of the Red-Black tree.
- **`printTotalWordCount`**: recursively computes and prints the total number of unique words currently stored in the tree (number of elements in the tree).
- **`printWordFrequencies`**: recursively prints each unique word in the tree in alphabetical order along with their frequencies in the text file.
- **`printMostFrequent`**: finds and prints the most frequent word with its frequency in the text file.
- **`printLeastFrequent:`** finds and prints the least frequent word with its frequency in the text file.
- **`printStandartDeviation:`** computes and prints the standard deviation of word frequencies in the text file.

The program must be compiled using a Makefile you provided and it should run with the following command. Whenever its computation is finished it should produce two output files, named as "**wordfreqs**" and "**statistics**". Sample execution will be as follows (As you can see input file name will be given as parameter in command line):

./redblackfreq <input_filename>

Output files will be produced in the directory your program executed and contain the following information:

wordfreqs: This file will contain each unique word with its frequency on each line. Words must be printed in alphabetical order.

Sample:

```
at 4
....
hello 22
...
yellow: 40
```

statistics: This file will contain statistics about input text file and the Red-Black tree in the following format:

Sample:

```
Total Word Count: 50
Tree Height: 8
Most Frequent: yellow 40
Least Frequent: at 4
Standard Deviation: 5,76
```

HAND-IN

- You should prepare the answers of 1) Question Part using a word processor (in other words, do not submit images of handwritten answers).
- Before 23:55 of July 31, 2019, upload your solutions using Moodle. You should upload a single zip file that contains
  - `hw4.pdf`, the file containing the answers to Question 1 & about scalability in Question 2.
  - Code files (only the "**.cpp**" and "**.h**" files that you write for the homework) of the second part (Question 2) and a "**Makefile**" for the compilation of your code that produces the executable.
  - `readme.txt`, the file containing anything important on the compilation and execution of your program in Question 2.
  - Do not put any unnecessary files such as the auxiliary files generated from your favorite IDE. Be careful to avoid using any OS dependent utilities.
  - Do not forget to put your name and student id in all of these files. **Well comment** your implementation.
  - Name your zip file as follows: **`NAME_SURNAME_hw4.zip.`**
  - If you do not know how to write a Makefile, you can find lots of tutorials on the Internet. Basically they are files that contain a list of rules for building an executable that is used by "make" utility of Unix/Linux environments. "make" command should build an executable called "**redblackfreq**", so write your Makefile accordingly (i.e. at the end, when you type "make" in terminal on your working directory, it should produce "**redblackfreq**" executable).
  - Keep all the files before you receive your grade.
  - The test file is available on the course website next to this document. It is James Joyce's *Dubliners* (https://en.wikipedia.org/wiki/Dubliners). Please note that your program will be tested with additional data.
  - This homework will be graded by your TA, Elmira Khajei (elmira.khajei AT bilkent edu tr). Thus, you may ask your homework related questions directly to her.

IMPORTANT:

- Although you may use any platform or any operating system to implement your algorithms and obtain your experimental results, your code should work on the **dijkstra** server (dijkstra.ug.bcc.bilkent.edu.tr). We will compile and test your programs on that server. If your C++ code does not compile or execute in that server, you will lose points.
- **Attention:** For this assignment, you are allowed to use the codes given in our textbook and/or our lecture notes and slides. However, you ARE NOT ALLOWED to use any codes from other sources (including the codes given in other textbooks, found on the Internet, belonging to your classmates, etc.). Furthermore, you ARE NOT ALLOWED to use any data structure or algorithm related function from the C++ standard template library (STL).

Do not forget that plagiarism and cheating will be heavily punished. Please do the homework yourself.