Lab 9 (Binary Tree, HashMap, and Priority Queue)

CSC 172 (Data Structures and Algorithms)
Fall 2017
University of Rochester
Due Date: Sunday, Nov 19 @ 11:59 pm

Introduction

The labs in CSC172 will follow a pair programming paradigm. Every student is encouraged (but not strictly required) to have a lab partner. Every student must hand in their own work but also list the name of their lab partner (if any) on all labs.

Objectives

This lab is designed to give you a chance to play with Binary Trees, HashMap, and Priority Queues. Also, it should aid you to get started with Project 3 if you have not done so yet. In the last lab, you have provided the inorder and pre-order traversal for a given tree. In this lab, you will do simply the opposite. Given an inorder and a pre-order traversal, you have to reconstruct the tree. In Lecture 18, we have covered HashMap and Priority Queue. The last two problems will give you a chance to explore these two very important data structures.

Problem 1: Level-order printing

Assume the following BTNode class is given.

```
public class BTNode
{
  int data;
  BTNode left;
  BTNode right;
 // Add constructor and/or other methods if required
}
```

Write a program that prints level-order traversal in a way that nodes of all levels are printed in separate lines. Method signature:

```
void level_order_print(BTNode root);
```

```
      1
    /    \
  2      3
/   \      \
4    5      6


Output for above tree should be
1
2 3
4 5 6
```

Problem 2: Reconstructing a Tree

Write a program that reconstructs a tree from its inorder and preorder traversal (refer: Lecture 19)

Method signature:

```
BTNode reconstruct_tree(int[] inOrder, int[] preOrder);
```

Now, perform in-order, pre-order, and level-order traversal on the resultant tree to verify the result. You can use code from your last lab and Problem 1 for this purpose.

Problem 3: Word-count problem

In this problem, you will read each word in a text file and report the number of times each word has appeared in the file. Instead of printing this information on the console, you will write the output to another file.

You must have to use HashMap to solve this problem.

To test your function, you should try the following steps:
First, download the tar files provided in Project 3 and extract the file. You will find a text file, 'alice30.txt' in the extracted folder. Count the number of times each word has appeared in the file (assume, each word is separated by only default delimiter set. To achieve this, read one line at a time and then split on "\\s+".). You do not need to ignore cases (i.e., consider, Alice and alice as two different strings). A sample entry in the output file may look like: Alice:221

Method signature:

```
void word_count(String inputFile, String outputFile);
```

Problem 4: Jesse's cookies

Adapted from: `https://www.hackerrank.com/challenges/jesse-and-cookies/problem`

Jesse loves cookies. He wants the sweetness of all his cookies to be greater than value K. To do this, Jesse repeatedly mixes two cookies with the least sweetness (sweetness is never negative). He creates a special combined cookie with:

sweetness = 1* sweetness of Least sweet cookie + 2 * sweetness of 2nd least sweet cookie).

He repeats this procedure until all the cookies in his collection have a sweetness $\geq$ K.

Write a function Jesse_cookies() that takes an integer array containing the sweetness of original cookies and a number K representing the minimum required sweetness. Return the number of operations (mixing) required to give the cookies a desired sweetness. Return -1 if this is not possible.

Method signature:

```
int Jesse_cookies(int[] cookies, int k);
```

> Note: We are not going to use HackerRank for submission this week. Please check the submission guidelines.

Submission

Save the four files as `Lab9P1.java, Lab9P2.java, Lab9P3.java, and Lab9P4.java`. Hand in these four files and a ReadMe file at the appropriate location on the Blackboard system at `learn.rochester.edu`. You should hand in a single zip (compressed archive) `Lab9zip`. The ReadMe file should state your and your team member's name and any other pertinent information.

Grading (100 pts)

NO LAB DEMO. Each problem is worth 25 points.

We will keep on updating this section to address your questions. Please check the lab description periodically.

- Each file must contain a main() method that calls the respective method for the problem.
- Fixed typo: To achieve this, read one line at a time and then split on "\\s+".
- Answer to Piazza post: `https://piazza.com/class/j6it5d7tcv43eg?cid=237`

```
Fixed. It should be \\s+.
(Sorry for the typo. LaTex ignores \ character).
[You are welcome to use [^a-zA-Z], but it's not a requirement.
It's fine if you keep special characters as it is.
That means: Alice, Alice's, Alices, are three different words.
We are more concerned about the algorithm than the actual output.]

For the last part:
You should have a main method which calls your function
(otherwise, how will you test your code yourself?).
TAs may or may not use the same main method that you wrote.
```