School of Computing
National University of Singapore
CS2010 Data Structures and Algorithms 2
Semester 2, AY 2015/16

Lab 1 – Searching with a Binary Search Tree and Linked List – Quantitative Analysis

Assigned – Jan 19th, 2016
Lab – Jan 25th, 2016 (please try to complete before the Lab. The lab TA will also provide assistance in the lab. Lab will be checked during the lab period. Please attend your assigned Lab). Code to IVLE must be submitted by the end of the day on Jan 25th (Monday).

Objective

To under search time complexity difference between linked list and binary search tree through empirical measurement.

Preparation:

Download the file lab1.zip from IVLE into your working directory. Uncompress the file and you should find the following document, java source code and data files: **CS2010-Lab1.pdf**, **LinkedList_BST.java**, **BST.java**, **LinkedList.java** and **10 data files**.

The BST code is from the class notes. The Linked List code is from another source. Both allow objects to be inserted. Read the code of the linked list (LinkedList.java) and binary search tree (BST.java) to understand how to implement those two data structures and read the main method (LinkedList_BST.java) to know how to use them.

Task:

Your task is to modify the associated search methods for the Linked List and BST code provided to you such that you can record the number of comparisons made by your List and BST objects. This will allow you to get real quantitative statistics about your data structures. You only need to record this when **searching** for a Key/Object in the tree, **not for inserting** Keys/Objects in the tree. You should also keep track if the item was found (HIT) or not found (MISS) when doing the search.

You are to print the statistics observed for each of the objects: 1) Sorted Linked List, 2) BST without random shuffle; and 3) BST with random shuffle.

Specifically, print the following:

```
Elements Found (HIT)
Linked list                 : Average comparison time = ?? <- You need to compute these
BST (without shuffling data): Average comparison time = ??
BST (with shuffling data)   : Average comparison time = ??
Elements Not Found (MISS)
Linked list                 : Average comparison time = ??
BST (without shuffling data): Average comparison time = ??
BST (with shuffling data)   : Average comparison time = ??
Overall (HIT + MISS)
Linked list                 : Average comparison time = ??
BST (without shuffling data): Average comparison time = ??
BST (with shuffling data)   : Average comparison time = ??
n      = 10000
log(n) = 14
```

【Input】A data file and a probing data file (stored in directory /data/*)

<*number*>*.dat* is a data file whose number means the number of whole data. *probe_*<*number*>*.dat* is a probing data file which corresponds to the data file with same <number>. For example, *1000.dat* is a data file which has 1000 integers and *probe_1000.dat* is the corresponding probing data file.

Most Java IDE will require the "data/" to be at the same level as the project files.

【Output】Number of comparisons happened in searching

You need to print average comparison times for probing data in linked list and binary search tree and print average comparison times when searching hits and misses in linked list and binary search tree separately.

【Source Code Provided】Java code from notes and other sources

The BST for this Lab comes from the class notes, please pay attention to the use of generics. The Linked List is from another source that does not use generic types. As a result, you do not need to pass types, however, you do need to cast when you use linked list (see code). In contrast, you must pass types when you use the BST which is a generic class. You can see the difference of classes with and without generics.

Submission Instruction (Lab Demo + IVLE submission)

You should demo the lab in class, but also please submit your code to IVLE:

1. Submit the softcopy of your Java codes to IVLE.
2. Please put your Java codes in a folder with the source files and submit the entire folder zipped. Use the following convention to name your zipped folder: MatriculationNumber_yourName_Lab1. For example, if your matriculation number is A1234567B, and your name is Chow Yuen Fatt, for this assignment, your file name should be A1234567B_ChowYuenFatt_Lab1.zip.
3. It is up to you to test to make sure that your Java code in the zipped file is complete. It is recommended that you unzip your Java code in a different location and test it to make sure it runs.