Homework 5

(Maximum 50 points)

Due beginning of class Wednesday October 14, 2015

Show the steps of deriving your answers.

1. **(50 points) [Tree performances: programming]** Implement three types of trees -- binary search tree, AVL tree, splay tree -- and compare their performance in the following steps.

Step 1 (Tree construction): For each type of tree, construct a new tree by inserting 100,000 nodes. Construct two trees for each type -- one by inserting keys in an *increasing* order and one by inserting keys in a *random* order. For each type of tree, report the total running time in each case of key ordering (i.e., increasing or random).

Step 2 (Tree searching): For each type of tree constructed with randomly ordered keys in Step 1, search the tree to find nodes 50,000 times with random key values. Do this twice -- first using each key exactly once and, second, using each key repeatedly 100 times. (So, 50,000 random keys are needed in the first key set, and 500 random keys are needed in the second key set. The total number of search operations performed is 50,000.) Report the total running time for each type of tree with each search key set.

Discuss your observations on the running time with an explanation. Use elapsed time (i.e., clock time in milliseconds or finer) as the running time. Feel free to utilize codes in the textbook or any other source (cite it in that case) – textbook codes are available at http://users.cis.fiu.edu/~weiss/dsaajava3/code/. Hand in your discussion in class and submit the source codes separately via Blackboard.

Programming tips:

- We can use the static function "System.nanoTime()" to measure the elapsed time in nanoseconds, as shown below.

```
long begin = System.nanoTime();
... //add here the code for which you want to measure the elapsed time.
long end = System.nanoTime();
long elapsed_time = end - begin;
```

- If a stack overflow error occurs (due to "too deep" nesting), we can increase the stack size using the Java option "–Xss stack_size", for example, "java –Xss 1024k MyProgram" (for a stack size of 1Mbytes).

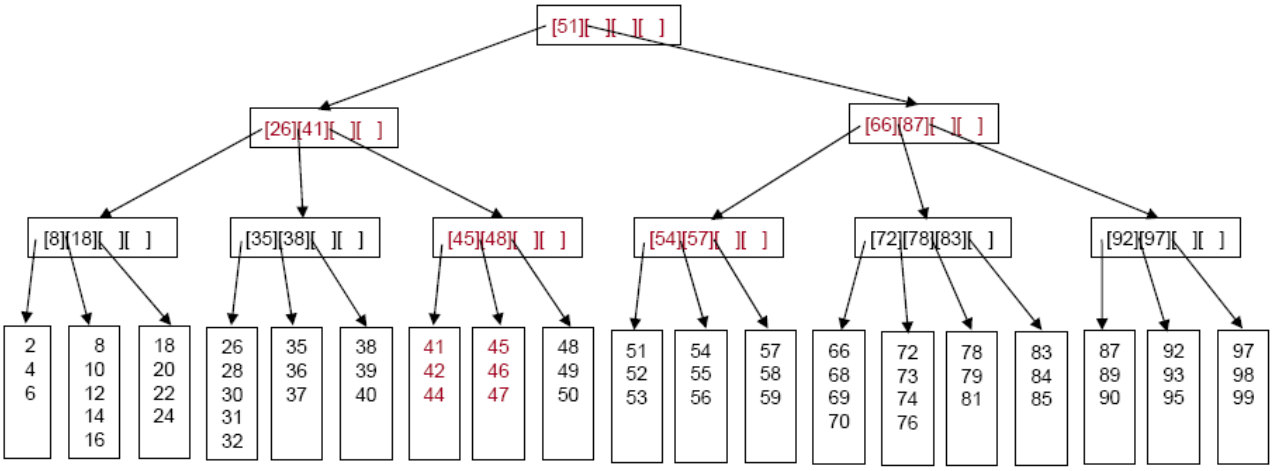
2. (0 point) [B-tree: insertion] -- *answer key provided below.*

a) Insert the keys 45 and 47 to the B-tree shown in Figure 4.63 of the textbook and show the resulting B-tree structure.

b) How many disk page accesses (i.e., read or written) occur while inserting the key 47 (after inserting the key 45) in the exercise b? Give the total number and explain it.

Answer key to Exercise 2:

a)



- No node split occurs as a result of inserting 45 into the node [41, 42, 44, 46].
- Node split occurs as a result of inserting 47, and the split propagates upward. The split nodes are shown in red color.

*b) 14 disk page accesses. There are 3 node-splits (i.e., data node and the first and the second level tree nodes) and 1 node (i.e., the new root) addition during the insertion of 47. Each node split incurs 4 disk page accesses (i.e., read-write for each of the two split nodes) and a new node addition incurs 2 disk page accesses (i.e., read an empty page and write a filled page). So, the total number of disk page accesses is 3 * 4 + 2 = 14.*

Last modified: September 30, 2015