Implementation of data structures and algorithms
Fall 2018
Short Project 4: Binary search trees
Thu, Sep 20, 2018

Version 1.0: Initial description (Thu, Sep 20).

Due: 11:59 PM, Sun, Sep 30.

Submission procedure:

* Create a folder whose name is your netid (NId).
* Place all files you are submitting in that folder.
* Use "package NId;" in all your java files.
* Include the class files also in your zip file.
* Include a text file named "readme.txt", that explains how to compile and run the code.
* Zip the contents into a single zip or rar file.
* If the zip file is bigger than 1 MB, you have included unnecessary files.
* Delete them and create the zip file again.
* Upload the zip or rar file on elearning.
* Submission can be revised before the deadline.
* The final submission before the deadline will be graded.
* Only one member of each team needs to submit project.
* Include the names of all team members in ALL files.

Team task:

1. Implement binary search trees. Starter code: BinarySearchTree.java.

Optional tasks (for individual submission):

2. Additional tasks on BST:
   Implement a bounded-sized stack using arrays with the operations push, pop, and isEmpty. Use it to implement iterator(), without copying the elements into another data structure like array or list. The problem can be solved using just O(h) extra space for stack of ancestors, where h is the height of the tree. In the iterator's constructor, find the height of the tree and allocate an array of size h for the stack. Implement floor(), ceiling(), predecessor() and successor() methods also.

3. Implement BSTMap (like a TreeMap), on top of BST class. Starter code: BSTMap.java

The following problems should be solved using TreeMap/TreeSet and other data structures in the Java library. Do not use hashing (HashMap/HashSet).

4. Given an array A of integers, and an integer X, find how many pairs of elements of A sum to X:

```
   static int howMany(int[] A, int X) { // RT = O(nlogn).
     // How many indexes i,j (with i != j) are there with A[i] + A[j] = X?
     // A is not sorted, and may contain duplicate elements
     // If A = {3,3,4,5,3,5} then howMany(A,8) returns 6
   }
```

5. Given an array A, return an array B that has those elements of A that occur exactly once, in the same order in which they appear in A:

```
   static> T[] exactlyOnce(T[] A) { // RT = O(nlogn).
     // Ex: A = {6,3,4,5,3,5}.  exactlyOnce(A) returns {6,4}
   }
```

6. Given an array A of integers, find the length of a longest streak of consecutive integers that occur in A (not necessarily contiguously):

```
static int longestStreak(int[] A) { // RT = O(nlogn).
  // Ex: A = {1,7,9,4,1,7,4,8,7,1}.  longestStreak(A) return 3,
  //    corresponding to the streak {7,8,9} of consecutive integers
  //    that occur somewhere in A.
}
```