TCSS 342 Spring 2017

Assignment 5
Due March 8

1. [**Heap**, 15 pts] Write a method in pseudocode that takes two heaps as inputs of size n and outputs a heap that contain all elements of the two given heaps. The heaps are given in a linked-list representation. The time complexity of your method should be $O(\log(n))$. Your method should be called public MergeHeap(Heap h, Heap g). You can assume that both heaps are perfect i.e. every level i contains 2^i nodes.

2. [**AVL**, 15 pts] Add a method called int subtreeSize(AVLNode N) to AVLTree.java class that takes a node N and return the number of nodes in the subtree rooted at N included N itself. The running time of your method should be O(1) in the worst-case. [Hint: Add another field called int size to AVLNode.java class and update size parameter during insertion].

3. [**ADT**, 30 pts] Consider the following abstract data type that supports the following operations

1. insert(int k): Insert a key k into the data structure only if it is not already in the ADT.

2. search_next(int k): Find the smallest key in the data structure that is greater than k.

3. search_smallest(int k): Find the kth smallest key in the data structure. For example search_smallest(0) will return the smallest key in the ADT.

   All operations are operations are done in $O(\log(n))$ where n is the number of keys in the data structure.

Implement the above ADT in java. Which data structure we saw in class is most appropriate to implement this ADT? You are allowed to use any java code you have been provided. You can assume the keys are of type int.

4. [**Hashing**, 40 pts] In this problem, you have to implement (in java) a hash map and observe how different load factor impact the performance of hash data structure. Collisions must be resolved by separate chaining.

- You have been provided a data set for this task is a CSV file (zipcode.csv) with two columns, namely zipcode and population.
- You have to run the program for load factor $\alpha = 0.85$, $\alpha = 0.80$, $\alpha = 0.75$ and $\alpha = 0.70$. Note the number of keys for this problem are fixed (see the data file). Therefore the table size is determined by the load factor.
- Your class should be called hashMap.java and must include the following methods.
  1. void insert(String key, String data) new entries in the table using Universal hash function(the one we saw in the class).
  2. int universalHash(String key): return the hash value of the key. Note keys are of type strings, thus they must be prehashed using the hashCode method for strings.
  3. String search(String key): return the population from the hash table for a given ZIP code.
  4. private void readCsv(). This function will read the CSV file and insert all the entries into the hash table. Note here keys are the ZIP codes.
  5. double average_length( ): Return the average length of the linkedlists. Note the average linkedlists length=$\frac{\text{size}}{\text{k}}$. Where k=number of non-null entries in the table and size is the sum of all the linkedlists lengths.
  6. double average_collision( ): return the average number of collisions. Note average_collision( )=average length-1. Note this should not be < 0.
- You program must also include a Main.java class. This class will only contain a main method that will run the hashMap.java class with different load factors $\alpha = 0.85$, $\alpha = 0.80$, $\alpha = 0.75$, and $\alpha = 0.70$, and print the average number of collisions for each α.