CMPS 1500 Introduction to Computer Science I – Fall 13

10/28/13

8. Homework

Programming portion due **11/5/13** at 11:55pm on Blackboard.
Written portion (part 4(b)) due **11/6/13** at the beginning of class.

Please create one or more Python files for this homework, and use the following naming convention: `lastName_firstName_hw8_Number.py`. The written portion can be turned in on paper.

In order to receive any credit for the programming portions, you are required to thoroughly comment and test your code.

1. **FIFO Queue (8 points)**
   A first-in first-out queue (*FIFO Queue*) is a data structure that conceptually stores a linear list of items by providing the following functionality:

   - `enqueue(item)` appends the new `item` at the end of the queue.
   - `dequeue()` removes the front item from the queue and returns it. It returns `None` if the queue is empty.
   - `isEmpty()` returns `True` if the queue is empty, and `False` otherwise.

   Write a class `Queue` that uses a linked list to store a FIFO queue, and that implements the three **methods `enqueue, dequeue, isEmpty` in constant time**. For this you have to store a reference to the `front`, as well as to the `rear` of the queue, and you have to store the `size` of the queue (the number of elements it contains). The three attributes/variables `front, rear, size` have to be initialized in the constructor `__init__`. Make sure that you implement `enqueue, dequeue, isEmpty` as methods (i.e., as part of the class) and not as functions (i.e., outside of the class).

   As comments in the code, justify why the runtime of `enqueue, dequeue, isEmpty` is constant.

2. **Linked list from BST (3 points)**
   Write a function that takes as input a binary search tree, and returns a linked-list that stores a post-order traversal of the tree.

3. **BST from array (4 points)**
   Write a function that takes as input a sorted array (i.e., python-list) `L` of numbers. The function should return a reference to the root node of a binary search tree that stores all numbers in `L`.
   (Hint: Use recursion. Repeatedly compute the median, similar to binary search, and use recursion to create the left and right subtrees.)

Flip over to back page ⟹

4. **EXTRA CREDIT: BST from linked list (8 bonus points)**

   (a) (4 bonus points) Write a function that takes as input a reference to the front node `front` of a linked list. This linked list contains a post-order traversal of a binary search tree of numbers. The function should return the binary search tree that has this post-order traversal.
   (Hint: Use recursion. The post-order traversal has the root at the very end, and the first part of the traversal consists of numbers less than the root, while the latter part consists of numbers greater than the root.)

   (b) (2 bonus points) Describe an example list that will cause the best-case runtime, and describe an example list that will cause the worst-case runtime.

   (c) (2 bonus points) What are the best-case and the worst-case runtimes of your function?