Data Structures

Fall 2019, Homework #2

Date: Oct. 28, 2019

1. (30 pts) *Queue from Two Stacks*
   A First-in First-out (FIFO) queue supports the following functionality: *enqueue(x)* adds element x to the end of the queue; *dequeue()* returns the element from the front of the queue. You will implement a QueueFromStacks class that implements the queue functionality using two stacks.

   (a) (20 points) A queue can be implemented using two stacks stackA and stackB as follows: In order to enqueue an item, push it onto stackA. In order to dequeue an item, pop the top item from stackB; but if stackB is empty, First pop all elements from stackA and push them onto stackB, and afterwards pop the top item from stackB. Implement a QueueFromStacks that uses two stacks to implement the enqueue and dequeue methods of a queue of integers this way. You will need the push and pop methods, and an additional isEmpty() method that returns true if the stack is empty. Specify the running times of the enqueue and dequeue operations of your solutions in Big-O.

   (b) (10 points) Consider the following sequence of operations, and draw pictures on how the queue and the corresponding stacks change: enqueue(1), enqueue(2), enqueue(3), enqueue(4), dequeue(), enqueue(5), enqueue(6), dequeue().

2. (30 pts) *Infix to Prefix Conversion*

   (a) (15 pts) Design an algorithm to convert Infix to Prefix. If needed, you may use the Infix-to-Post conversion discussed in class as a subroutine.

   (b) (15 pts) Apply your algorithm to the following Infix Expression:
   A + (B * C - (D / E - F) * G) * H
   Show the steps of your conversion in detail. That is, show the output as well as the contents of the stack when a character is read.

3. (20 pts) *Trees*

   (a) (10 pts) Draw all binary trees of 4 nodes.

   (b) (10 pts) Draw all ordered (not necessarily binary) trees of 4 nodes.

4. (20 pts) *Tree Traversal*

   (a) (10 pts) Is it possible to construct a unique binary tree from its inorder and preorder traversal sequences? How?

   (b) (10 pts) Apply your method to the following:
   Inorder: D B E A F C
   Preorder: A B D E C F