CPSC 3200: Algorithm Analysis and Advanced Data Structure

Summer 2013

Home Assignment 2

Please work on the following programming exercises from your Data Structure and Algorithms in Java, 5th edition textbook.

Question #1: (Programming)

R-3.4: Write a class that maintains the top 10 scores for a game application, implementing *add* and *remove* methods of Section 3.1.1, but using a singly linked list instead of an array.

Question #2:

R-3.5: Describe a way to use recursion to add all the elements in a n×n (two-dimensional) array of integers.

Question #3:

R-3.12: Describe a nonrecursive method for finding, by link hopping, the middle node of a doubly linked list with header and trailer sentinels. (Note: This method must only use link hopping; it cannot use a counter.) What is the running time of this method?

Question #4:

R-3.14: Draw the recursion trace for the execution of method ReverseArray(A,0,4) (Code Fragment 3.32) on array A = {4,3,6,2,5}.

Question #5: (Programming)

Write a swap method to swap two nodes x and y (and not just their con-tents) in a singly linked list L given references only to x and y. Repeat this exercise for the case when L is a doubly linked list. Which algorithm takes more time?

Question #6:

R-4.2: Give a pseudo-code description of the O(n)-time algorithm for computing the power function *p(x, n)*. Also draw the recursion trace of this algorithm for the computation of *p(2, 5)*.

Question #7:

R-4.14: Give a big-Oh characterization, in terms of n, of the running time of the Ex3 method shown in Code Fragment 4.6 (Page 195).

Question #8:

R-4.15: Give a big-Oh characterization, in terms of n, of the running time of the Ex4 method shown in Code Fragment 4.6 (Page 195).

Question #9:

R-4.21: Bill has an algorithm, find2D, to find an element x in an n×n array A. The algorithm find2D iterates over the rows of A, and calls the algorithm arrayFind, of Code Fragment 4.5, on each row, until x is found or it has searched all rows of A. What is the worst-case running time of find2D in terms of n? What is the worst-case running time of find2D in terms of N, where N is the total size of A? Would it be correct to say that Find2D is a linear-time algorithm? Why or why not?

Grading system for each question.

Description
Class name
Comments for the "author" – FirstName, LastName
Comments for the program description
Code and **code comments** following the **Javadoc** style