COMP 3761 Assignment 3

Due: Wednesday **Feb 04, 2015** at 6:30pm

1. Given the following two **BubbleSort** variations:

   (i) The version of BubbleSort that we have presented in class can be improved by exploiting the following observation: if bubble sort makes no exchanges on its pass through a list, the list is sorted and the algorithm can be stopped.

   (ii) **Cocktail sort** is a variation of bubble sort that sorts in both directions on each pass through the list, where the rightward pass shifts the largest element to its correct place at the end, and the following leftward pass shifts the smallest element to its correct place at the beginning.

   Solve the following problems for each variation of the Bubble Sort:

   (a) Provide pseudocode of the method. [2 points x 2]

   (b) Theoretically analyze the worst-case time efficiency class of each variation. [2 points x 2]

   (c) Implement the algorithm in Java. Count the actual number of element comparisons and item swaps performed when the algorithm is executed for any input array. [2 points x 2]

   (d) Test your solution with the input data given in the directory a3_input:
   Test files: IntegerArray_1000.txt, IntegerArray_10000.txt, IntegerArray_100000.txt
   Each test file represents one input integer array of unique numbers between 1 and 100,000 (inclusive) in arbitrary order. The total number of integers in each array is indicated by the number as part of the file name. For example, the test file IntegerArray_1000.txt contains exactly 1000 distinct integers.
   The format of each input test file is the same: each line contains one single integer number, where the ith row of the file indicates the ith entry of an array.

      i. Write a method that reads in all the values from the input files and constructs the 3 input arrays as per file format specification. You may initialize an array (or ArrayList) of String to hold the three given test file names.
      Note: in this Assignment, you can hard-code the given file names, but do NOT use any absolute file path in the code. You can simply copy the files to the same directory where your java code is located. Print both the number of element comparisons and the actual number of element swaps executed in each case. [2 points]

      ii. Test both variations of the Bubble Sort with all the input array data. Compare the actual number of element comparisons and swaps executed with the given test data. Comment your observations on the actual comparisons made. [2 points]

2. **Alternating disks** You have a row of $2n$ disks of two colors, n dark and n light. They alternate: dark, light, dark, light, and so on. You want to get all the dark disks to the right-hand end, and all the light disks to the left-hand end. The only moves you are allowed to make are those which interchange the positions of two neighboring disks. Design an algorithm for solving this puzzle. Provide a pseudocode and determine the number of moves the algorithm makes. [5 points]

3. From the textbook Page 121. Exercises 3.4 # 6.

   Consider the **partition problem**: given n positive integers, partition them into two disjoint subsets with the same sum of their elements. (Of course, the problem does not always have a solution.) Design an exhaustive search algorithm for this problem. Try to minimize the number of subsets the algorithm needs to generate. Provide the pseudocode and determine its worst-case running time efficiency class. [5 points]