CMPS 1500 Introduction to Computer Science I – Fall 13

10/7/13

6. Homework

Due **10/22/13** at 11:55pm on Blackboard.

Please create a separate Python file for problem 1 and problem 2 below, and use the following naming convention: `lastName_firstName_hw6_number.py`.

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

1. **Card Sorting (8 points)**
   When sorting playing cards, one often maintains a sorted sub-list of cards and repeatedly inserts another card into that sub-list at the correct position, until in the end all cards have been sorted.

   (a) (7 points) Implement the following algorithm to sort numbers in increasing order, which is inspired by sorting playing cards: Given an input list L of n unsorted numbers. Iterate through all the numbers in L, maintain the property that the first i numbers (with indices $0...i-1$) are sorted, and now insert the number with index i at the correct position, such that now the first $i+1$ numbers are sorted. Don't forget to comment and test your code.

   (b) (1 point) What is the asymptotic running time of your algorithm? Please write your answer as a comment in your code, together with a very brief justification.

2. **Bottom-Up Mergesort (8 points)**
   The goal of this exercise is to implement a "bottom-up" non-recursive mergesort algorithm. A single pass of this algorithm takes a list of lists as input, and then merges every two consecutive lists using the `merge` function we used in class, to produce a new list of lists. For example, for an input list `L= [6, 3, 4, 10, 9, 1, 2, 5, 7, 0, 8]` the algorithm would create the following lists:

   Phase 1: `[[6], [3], [4], [10], [9], [1], [2], [5], [7], [0], [8]]`
   Phase 2: `[[3, 6], [4, 10], [1, 9], [2, 5], [0, 7], [8]]`
   Phase 3: `[[3, 4, 6, 10], [1, 2, 5, 9], [0, 7, 8]]`
   Phase 4: `[[1, 2, 3, 4, 5, 6, 9, 10], [0, 7, 8]]`
   Phase 5: `[[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]]`

   (a) (2 points) Write a function `make_list_of_lists(L)` that takes an input list `L` and returns a list that consists of singleton lists of elements in `L`. For example, if `L=[6, 3, 4, 10]` then the result would be `[[6], [3], [4], [10]]`.

FLIP OVER TO BACK PAGE ⟹

(b) (2 points) As a first warmup, write a function `merge_bottom_up_one_pass(LoL)` that takes a list of lists `LoL` as input, and performs one pass of the algorithm by merging every two consecutive lists using the `merge` function we used in class, to produce a new list of lists.

(c) (2 points) As a second warmup, write a function `merge_bottom_up_two_passes(LoL)` that takes a list of lists `LoL` as input, and performs two passes of the algorithm.

(d) (2 points) Now, write a function `merge_bottom_up(L)` that takes a list of numbers `L` as input, computes the list of singleton lists, and then performs as many passes as necessary to return one completely sorted list. Test your code.