TCSS 343: Design and Analysis of Algorithms
Spring 2018, Homework #3
Due: Tuesday, May 8

Please submit a hard copy of your answers to the homework problems below. Staple all of your pages together (and order them according to the order of the problems below) and have your name on each page, just in case the pages get separated. Write legibly (or type) and organize your answers in a way that is easy to read. Neatness counts!

For each problem, make sure you have acknowledged all persons with whom you worked. Even though you are encouraged to work together on problems, the work you turn in is expected to be your own. When in doubt, invoke the Gilligan's Island rule (see the syllabus) or ask the instructor.

All assignments (written or programming) are due at the beginning of the lecture on the due date. You may turn 1 assignment in up to 1 lecture late. No assignments will be accepted *after* the beginning of the lecture *after* the due date. The reason for this is that in the lecture after the due date, I might already return graded homeworks to students who submitted on time.

Regular problems (to be turned in):

1. (5 points) This problem refers to the pseudocode for *Quicksort* on p. 176 of the textbook. Assuming that the pivot is chosen as the item at index l, i.e. at the left most position, illustrate the behavior of *Quicksort* to sort the entries of the array A = [U,W,T,A,C,O,M,A] in alphabetical order. At a minimum, show the following: for each recursive call of *Quicksort*, show the contents of the array A from index l to index r. Arrange the calls into the nodes of a tree such that left children represent the first recursive call and right children represent the second recursive call.

2. (5 points) Consider the problem of searching for genes in DNA sequences using Horspool's algorithm. A DNA sequence is represented by a text on the alphabet {A, C, G, T}, and the gene or gene segment is the pattern.

   (a) Construct the shift table for the following gene segment:

   ACATCTCA

   (b) Apply Horspool's algorithm to locate the above pattern in the following DNA sequence:

   GAGTAATCCTTCACTTCAAGGCCAGTCTTCACATCTCATCAGA

   Show where comparisons take place and where shifts occur.

3. (15 points) You have a list of n open intervals $(a_1, b_1), (a_2, b_2), ..., (a_n, b_n)$ on the real line. An open interval (a, b) comprises all the points strictly between its endpoints a and b, i.e., $(a, b) = \{x | a < x < b\}$. You need to find the maximal number of these intervals that have a common point. For example:

   - $(1, 3), (4, 5), (6, 9)$: none of these intervals have a common point, so the maximal number is 1
   - $(1, 3), (2, 5), (6, 9)$: the first two intervals have a common point, so the maximal number is 2
   - $(1, 3), (2, 5), (4, 9)$: the first two intervals have a common point, the last two intervals have a common point, but the first and the last interval have no common point, so the maximal number is 2

- $(1,5),(2,5),(4,9)$: now all three intervals have a common point so the maximal number is 3
- $(1,4),(0,3),(-1.5,2),(3.6,5)$: in this case the maximal number is also 3

Another way to look at it is to think of every interval as the time spent by a person at a meeting point. For instance (1,3) means that this person arrived at time 1 and left again at time 3; (2,5) means that this person arrived at time 2 and left again at time 5; etc. The problem is asking for the largest number of people that are together at the meeting point at any give time during the whole duration of the experiment.

Be careful: end points of the intervals do not count. For instance (1,3) and (3,4) do not have any point in common. Imagine that the first person left at time 3 through the back door, while the second person entered at time 3 through the front door, so they never met.

Design an algorithm for this problem with a *better than quadratic* time efficiency.

To be submitted with the hard copy of your solution for this homework:

(a) A description of your algorithm in low level pseudocode.

(b) A brief explanation of how and why your algorithm works.

(c) An analysis of the time efficiency of your algorithm.

To be submitted in electronic version on Canvas:

(d) An implementation of your algorithm in Java or Python, as either hw3.java or hw3.py (one file). Within this file, call your algorithm with input $(1,3),(4,5),(6,9)$ so that it is very clear to me how to call the algorithm. I will try it with different test cases.