CISC621 Algorithm Design and Analysis, Spring 2016
Homework set I, due Thursday, February 25, 7:00pm

Check the homework rules from the syllabus, including in the "homework sheet", for general homework details. Submit in 201 Smith Hall directly to Fanchao Meng or place on shelf marked CISC 621. Keep each solution separate. Start each problem on a separate sheet of paper, include your name, and — if a solution uses multiple sheets of paper — fasten sheets for one problem together. DO NOT fasten solutions to two or more problems together.

1. **Recurrences:**

   Argue that the solutions to the following recurrences are as indicated by using recursion trees. Note that you must attend to both upper and lower bound. The c is a constant. [For reference, problems of this type are in section 4.4 and it's exercises.]

   (a) For $T(n) = T(n/4) + T(2n/3) + cn$, show $T(n) = \Theta(n)$.

   (b) For $T(n) = T(n/4) + 2T(3n/8) + cn$, show $T(n) = \Theta(n \log(n))$.

   (c) For $T(n) = T(n/2) + T(2n/3) + cn$, show $T(n) = \Omega(n^{1+ \log_2(7/6)})$. and $T(n) = \mathrm{O}(n^{1+ \log_{3/2}(7/6)})$.

   [You may (or may not) find it useful to note that $1 + \log_2(7/6) = \log_2(7/3)$ and $1 + \log_{3/2}(7/6) = \log_{3/2}(21/12)$ The intent of the exercise is that you can get both upper bound and lower bound by the recursion tree / work at level technique. By the way, $n^{1.22} < n^{\log_2(7/3)} \ll n^{\log_{3/2}(21/12)} < n^{1.39}$.]

2. **Glitch discovery**

   (a) The n children in music school are each assigned an identification number in the range 1..n. Likewise each of the n seats in the auditorium is assigned a unique number in range 1..n. Array A stores the seat assignments so that A[i] is the seat number for child i. Occasionally a mistake is made in seat assignments so that two children have the same seat assigned and some seat goes unassigned. Solve the problem of determining if a mistake has been made. You may assume at most one mistake has been made (i.e. one missing number and one duplication). You algorithm should run in O(n) time, not change the array, and use a constant amount of additional memory (i.e. some variables, but no second array). Explain why your algorithm is correct and runs in worst case time $\Theta(n)$.

   (b) In this case, the array of length n has all the numbers 0..n in it (in some random order) with one missing. Find the missing number. However, here you can only access the array with the function getBit(A,i,j) which returns the j-th bit in the binary representation of A[i]. For example if $A[i] = 13 = 1101_2$ then getBit(A,i,0) returns 1, and getBit(A,i,1) returns 0. Design and explain an algorithm to find the missing number. It must use O(n) calls to getBit in the worst case.

3. **Re-sort**

   We will have a problem later in which points (x,y) in the plane must be first sorted by x coordinate and later sorted by y coordinate. This problem is to sort by y coordinate an array of points that is already sorted by x coordinates, and do it in linear time. More specifically, on input array A of n points is sorted *lexicographically*. For lexicographic order, (x_1, y_1) is considered less than (x_2, y_2) if and only if either $x_1 < x_2$ or ($x_1 = x_2$ and $y_1 < y_2$). On output the array is to be sorted *reverse lexicographically*. For reverse lexicographic order, (x_1, y_1) is considered less than (x_2, y_2) if and only if either $y_1 < y_2$ or ($y_1 = y_2$ and $x_1 < x_2$). Explain your algorithm for correctness and for O(n) runtime.

   You may assume that all x and y coordinates are integers in the range 1..n. This allows you to do things like make a pass through A building an array N in which N[y] records the number of points having y as second coordinate. Alternatively, you may want to use an array of linked lists, see section 10.2 of CLRS (our textbook).