COMP 3711 – Design and Analysis of Algorithms
2019 Spring Semester – Written Assignment # 1
Distributed: Feb 20 2019 – Due: March 1, 2019

Your solutions should contain (i) your name, (ii) your student ID #, and (iii) your email address
Notes:

- Please write clearly and briefly.
- Please follow the guidelines on doing your own work and avoiding plagiarism given on the class home page.
  In particular ***don't forget to acknowledge individuals who assisted you, or sources where you found solutions.*** Failure to do so will be considered plagiarism.
- Also follow the submission guidelines on the class web page. Use white, unwatermarked paper, e.g., no student society stationary.
  - If handwritten, solutions should be single sided, start a new page for every problem, be single column and leave space between consecutive lines and more space between paragraphs.
  - If typed, try to use an equation editor, e.g., latex , the equation editor in MS-WORD or whatever the equivalent is in whatever typesetting system you are using
- This assignment is due by 13:00 on March 1, 2019 in BOTH hard AND soft copy formats. A hard copy should be deposited in one of the two COMP3711 assignment collection boxes outside of room 4210. A soft copy for our records in PDF format should also be submitted via the online CASS system. See the Assignment 1 page in Canvas for information on how to submit online.
- The default base for logarithms will be 2, i.e., $\log n$ will mean $\log_2 n$. If another base is intended, it will be explicitly stated, e.g., $\log_3 n$.

Problem 1 [21 pts] For each pair of expressions (A, B) below, indicate whether A is O, Ω, or Θ of B. Note that zero, one, or more of these relations may hold for a given pair. List all correct ones. It often happens that some students will get the directions wrong, so please write out the relation in full, i.e., $A = O(B), A = \Omega(B)$, or $A = \Theta(B)$ and not just $O(B)$, $\Omega(B)$ or $\Theta(B)$, omitting the A.

(a) $A = n^4 - n^3, B = 1000n^3 + 100n^2 + 10n$;

(b) $A = \log_9 4n, B = \log_{4.7} n^3$;

(c) $A = 50n \log n \log \log n, B = 2^n$;

(d) $A = (\sqrt{2})^{\log n}, B = \sqrt{\log n}$;

(e) $A = 2^{3 \log_3 n}, B = 47n^3$;

(f) $A = 2^{\log_2 n^5}, B = 6n^5$;

(g) $A = \ln \log n, B = 10^{100}$
B, which is 1 followed by 100 zeros, is called a "googol".

Problem 2 [25 pts] Give asymptotic upper bounds for $T(n)$ satisfying the following recurrences. Make your bounds as tight as possible. A correct answer will gain full credits. It is not necessary to show your work BUT, if your answer was wrong, showing your work steps may gain you partial credits. If showing your work, you may use theorems shown in class or can prove results from scratch.
If you do prove your results from scratch, for (a), (b) you may assume that n is an integral power of $5/3$; for (c), (d), (e) you may assume that it is an integral power of 3.

(a) $T(1) = 1; T(n) = T(3n/5) + 1$ for $n > 1$.

(b) $T(1) = 1; T(n) = T(3n/5) + n$ for $n > 1$.

(c) $T(1) = 1; T(n) = 9T(n/3) + n^2$ for $n > 1$.

(d) $T(1) = 1;\ T(n) = 7T(n/3) + n^2$ for $n > 1$.

(e) $T(1) = 1;\ T(n) = 5T(n/3) + n$ for $n > 1$.

Problem 3: [24 pts] For each of the following two problems, separately, give an asymptotic upper bound for $T(n)$ satisfying the associated recurrence. Make your bounds as tight as possible. You must prove these bounds from scratch (either using the expansion method or the tree method). You may assume that n is a power of 2.

(a)
$$T(1) = 1; \quad T(n) = 4T(n/2) + n^2 \text{ for } n > 1$$

(b)
$$T(1) = 2; \quad T(n) = 5T(n/2) + n^2 \text{ for } n > 1$$

Problem 4 [30 pts]

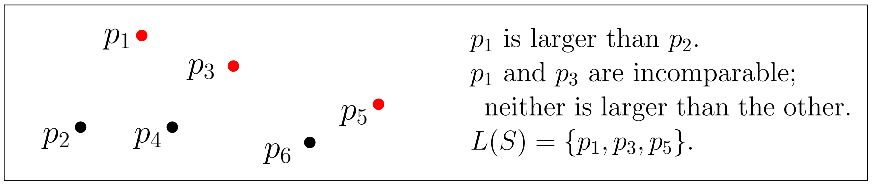
In two dimensions, point $p = (x, y)$ is *larger* than point $p' = (x', y')$ if

$$p'.x < p.x \quad \text{and} \quad p'.y < p.y.$$

Let $S = \{p_1, \ldots, p_n\}$ be a list of n two-dimensional points $p_i = (x_i, y_i)$. Point $p_j \in S$ is a *largest* point in S if no other point in S is larger than p_j.

$L(S)$ is the set of largest points in S, i.e.,

$$L(S) = \{p_j \,:\, \text{for all } i \neq j,\ p_i \text{ is not larger than } p_j\}.$$



p_1 is larger than p_2.
p_1 and p_3 are incomparable; neither is larger than the other.
$L(S) = \{p_1, p_3, p_5\}$.

The example in the box illustrates the definitions.

In this problem you should design an $O(n \log n)$ divide-and-conquer algorithm for finding $L(S)$, given input S. Do this following the four steps on the next page. Each step should be labelled and answered separately.

For simplicity, you may assume that no two input points share the same x-coordinate or the same y-coordinate.

Continued on next page....

1. Let $S^1 = \left\{p^1_1, p^1_2, \ldots, p^1_{n_1}\right\}$ and $S^2 = \left\{p^2_1, p^2_2, \ldots, p^2_{n_2}\right\}$ be two sets of points such that for both $i = 1, 2$, S^i has the following properties:

   (a) $L(S^i) = S^i$, i.e., every point in S^i is a largest point in S^i.

   (b) Each S^i is sorted by x-coordinate, i.e., $p^i_1.x < p^i_2.x < \cdots < p^i_{n_i}.x$.

   Describe an $O(n_1 + n_2)$ time procedure $Merge(S^1, S^2)$ that returns $L(S^1 \cup S^2)$, the set of largest points in the union of the two sets. The returned set should be sorted by x-coordinate.

2. Explain why your procedure $Merge(S^1, S^2)$ is correct and why it runs in $O(n_1 + n_2)$ time.

3. Using $Merge(S^1, S^2)$ as a subroutine, describe a $O(n \log n)$ time Divide-and-Conquer algorithm $FINDL(S)$ that returns $L(S)$.
   The returned set should be sorted by x-coordinate.

4. Explain why your procedure $FINDL(S)$ is correct and why it runs in $O(n \log n)$ time.

By convention, a set of points will be stored in an array S where $S[i] = p_i$ and $size(S) = n_i$. You may assume that arrays can be created in constant time and can be passed by reference by subroutines. Your procedures should return an array containing the result points, sorted by x-coordinate.

Note that the original input to the algorithm is arbitrary and not necessarily sorted by x-coordinate.

In parts 1 and 3 the description should be in clearly documented pseudocode.