CSC 425/520 Fall 2016: Assignment #1: Written Questions
Due at beginning of class, Tues. Sept. 27, 2016

Read Chapters 1-2 and Section 1.3 of the text (these sections review mathematics and algorithm analysis prerequisites).

1. Put your name on your assignment. Questions should be **in order**.
2. The written questions should be submitted on paper at the beginning of class on the due date. Submissions can be handed in up to 4 days late with a 10% penalty for each day past the deadline. If you want to submit an assignment on a day that we do not have class, please write the day and time submitted on your first page and slip it under my office door (ECS 552).
3. Show your work unless otherwise stated.
4. Draw BIG boxes for your marks on the top of the first page of your submission. Place a 0 in the corresponding box for any questions you omit. For this assignment:

Question:	1	2	3	4	5	6	7
Marks	0	0	0	0	0	0	0

1. A student counted up the number of statements executed in a program developed for sorting n integers and came up with a recurrence of the form:
   $T(n) = a * n + b + T(n-1)$, and $T(1) = s$ where a, b and s are constants.
   (a) [2] Use the method of repeated substitution to solve this recurrence.
   Your solution should be a closed formula: sums or expressions with T in the formula are not permitted.
   (b) [2] Prove your answer to (a) is correct by induction.
2. The purpose of this question is to show that as long as a, b and s from question 1 are constants satisfying $a > 0$, and $b, s \geq 0$, the final answer for the recurrence always has the same Big Oh time complexity. This rationalizes making the recurrences we solve for time complexities as simple as possible when doing algorithm analysis. For example, $T(n) = n + T(n-1)$, and $T(1) = 1$ is a simpler recurrence giving the same time complexity ($a = 1$, $b = 0$, and $s = 1$).
   (a) [3] Assume that $a > 0$, and $b, s \geq 0$. Find constants n_0 and c such that for all $n \geq n_0$, your answer from question 1 is at most $c \cdot n^2$. Hint: you will have to express your constant c as some function of the constants a, b and s.

(b) [3] Assume that $a > 0$, and b, $s \geq 0$. Find constants n_0 and d such that for all $n \geq n_0$, your answer from question 1 is at least $d \cdot n^2$ for some $d > 0$.

3. Consider a quicksort algorithm that operates on linked lists as follows:
If the problem size is 7 or less a MaxSort algorithm is used to sort the values.
Otherwise, let x be the data value in the first linked list node.
Rearrange the pointers of the original list to create three new lists:
P1: values less x.
P2: values equal to x.
P3: values greater than x.
Sort P1 and P3 by calling quickSort recursively.
Return P1 followed by P2 followed by P3.

(a) [5] Why is this recurrence relation a reasonable one to use in order to analyze the best case time complexity of this algorithm **under the assumption that all the data values are distinct:**
$n = 2^k - 1$ for some integer k, $T(7) = 23$, $T(n) = (n + 1) + 2T((n - 1)/2)$.

(b) [5] Solve the recurrence relation from (a) using repeated substitution. Hint: it is easier to express things in terms of k then to work with n directly. Convert your answer back to an expression in terms of n when you are done solving the recurrences.

(c) [5] Prove that your answer to (b) is correct by induction. Or for part marks, continue an induction proof until it fails (if your answer to (b) is not correct).

4. Consider these three recurrence relations that are defined for $n = 2^k$ for some $k \geq 0$:

$$L(n) = n^2 + 2L(n/2),\ L(1) = 1.$$
$$T(n) = 4n^2 + 3n + 2 + 2T(n/2),\ \mathrm{T}(1) = 4.$$
$$U(n) = 12n^2 + 2U(n/2),\ U(1) = 12.$$

For parts (a), (b), (c), and (d), answer these questions directly (without solving for a solution to the recurrence relations).

(a) [5] Prove by induction that U(n) = 12 * L(n).

(b) [5] Prove by induction that L(n) < T(n) < U(n).

(c) [3] Use your answers to parts (a) and (b) to prove that $T(n) \in \Theta(L(n))$.

(d) [2] If the actual number of operations is given by T(n), does it make sense to solve for L(n) instead if the asymptotic time complexity is desired? Justify your answer.

5. [5] Take the following functions and arrange them in ascending order of growth rate. That is, if function $g(n)$ immediately follows $f(n)$, it should be the case that $f(n) \in O(g(n))$.

$$f_1(n) = 100\,n^3 + 25n^2 + 13n + 42$$
$$f_2(n) = n^3\sqrt{(}n)$$
$$f_3(n) = n^3 \log_2(n);$$
$$f_4(n) = n^{\log_2(n)}$$
$$f_5(n) = 2^{2\log_2(n)}$$
$$f_6(n) = 2^n$$
$$f_7(n) = 2^{n\log_2(n)}$$
$$f_8(n) = 2^{987654321}$$

6. Dominating sets of trees.

   (a) [5] Give pseudocode for a polynomial time algorithm for finding **a minimum dominating set** for a tree.

   (b) [5] Analyze the time complexity of your algorithm.

7. The solution to this question gives a simpler proof that Dominating set is NP-complete than the one in class. A subset S of the vertices of a graph G is called a *Dominating Set* if every vertex v of G is either in S or v has a neighbour which is in S (that is, there is some vertex u in S such that (u, v) is an edge of G). Given a 3-SAT problem, construct a graph as follows: For each variable x, the graph contains a triangle whose 3 vertices are labelled with x, $\bar{x}$, and x' (x and $\bar{x}$ correspond to literals). For each clause, there is a vertex of the graph connected to the three literals in the clause. If the 3-SAT problem has k variables and c clauses then this constructed graph has $3k + c$ vertices and $3k + 3c$ edges.

   (a) [2] Draw the graph that arises from this 3-SAT system:
   $(x \text{ or } y \text{ or } z)$ and $(x \text{ or } y \text{ or } \bar{z})$ and $(x \text{ or } \bar{y} \text{ or } z)$ and
   $(x \text{ or } \bar{y} \text{ or } \bar{z})$ and $(\bar{x} \text{ or } y \text{ or } z)$ and $(\bar{x} \text{ or } \bar{y} \text{ or } \bar{z})$

   (b) [3] Find a satisfying assignment of this 3-SAT system and show a corresponding dominating set on the graph.

   (c) [5] For a general problem, if the 3-SAT system has a satisfying truth assignment, what size dominating set can you find?

   (d) [5] Prove that it is not possible to find a dominating set of the size you specified for part (c) when the 3-SAT system has no satisfying truth assignments.