Comp 160: Algorithms, **Recitation 1**

1. Consider the recurrence $H(n) = 2H(n/2) + n$. Quickly review what steps would you follow to show that $H(n) = O(n \log n)$ using substitution and recursion tree

2. State the master theorem, clearly marking the three possible cases and their solution. Which case applies to $H(n)$ (described above)?

3. In lectures we discussed the recurrence $G(n) = 2G(\frac{n}{4}) + n \log n$

   (a) Repeat the process done on class to prove an upper bound using recursion tree

   (b) What is the height of the tree? What is the cost at the top level? does the cost grow or shrink at each level?

   (c) Using the master method, how many leaves does the tree have? How much do we pay at each leaf node? what is the level sum of the bottommost level?

   (d) Give an $\Omega(n \log n)$ BUT you are not allowed to use neither substitution nor recursion tree nor the master theorem.

   Hint: look again at the recursion and think what it does. Proof itself is two lines.

   (e) Finally, use the master theorem to prove matching upper and lower bounds for $G(n)$

4. Use the master method, if possible, for the following recurrences:

   (a) $T(n) = 16 \cdot T(\frac{n}{4}) + n^2$

   (b) $T(n) = 256 \cdot T(\frac{n}{4}) + \Theta(n^4 \log^4 n)$

   (c) $T(n) = 157 \cdot T(\frac{n}{157}) + n^2$

   (d) $T(n) = T(\frac{159}{732}n) + \Theta(n^2)$

5. Let's use recurrences to give bounds on the runtime of binary search:

   (a) First quickly describe the goal of algorithm, its requirements and write its pseudocode in a recursive fashion

   (b) Give a recurrence that describes its worst case runtime and justify it (with 1 sentence)

   (c) Draw the recursion tree. How many levels does it have? What is the cost at the top level? What about the bottom level?

   (d) Use the answer in the previous sections to give a lower bound

   (e) Use substitution to prove a matching lower bound

Additional practice questions:

6. Consider the recurrence $T(n) = \pi T(\frac{n}{7}) + \frac{4n}{5}$

   (a) Draw 2-3 levels of the recursion tree.

(b) What is the cost of those three levels? Is the cost growing or shrinking?

(c) Now that you have an estimate, find upper and lower bounds using substitution

Hint: the math is very ugly, but don't let it scare you. From an algorithmic point of view, it does not make much sense to say that an algorithm makes π recursive calls...but once you see past the numbers, you will see that it is solved exactly like the ones explained in class. You can actually use the master theorem to verify your estimate is correct.

7. Design a recursive algorithm that sorts n numbers and whose runtime is $T(n) = 3T(2n/3) + O(1)$. What is the runtime of this algorithm?

8. Let $T(n) = 2T(n/2) + f(n)$. Pick three different values of $f(n)$ so that for each of them different cases of the Master Theorem applies. Can you do the same process for $G(n) = 5G(n^2/6) + f(n)$? Why?

9. Consider the following sorting algorithm, for an array of n distinct elements, $a_1, \ldots, a_n$.

   The algorithm does a *scan* of the array as follows: iterate i from 1 to n. At each step of the iteration, compare a_i with a_{i+1} and swap them (if necessary) so that a_i is smaller. Continue executing scans of the array until a scan creates no swaps.

   Answer with a **brief** argument (your answer for each letter should be a short paragraph):

   (a) First let's focus on termination and correctness. What is accomplished in the j-th scan of the array? How do we use this to prove that the algorithm terminates and will always give the correct result (that is, have the array sorted)?

   (b) What is the worst-case runtime of the algorithm? Hint, focus in the number of comparisons

   (c) How could you cut the worst-case overall number of comparisons in half?

   (d) How does the algorithm perform on an array that is already sorted, or reverse sorted?

   (e) What if the array is already sorted except that a_1 is the largest element (and all other elements are shifted one position to the right)? What if the array is already sorted except that a_n is the smallest element (and other elements are shifted to the left)? How does this compare to Insertion sort?

10. **Challenge:** Consider the recurrence $T(n) = T(\sqrt{n}) + \log n$

   (a) Find an upper bound for $T(n)$ with a recursion tree.

   (b) Find a lower bound for $T(n)$ by substitution.

   (c) Solve $T(n)$ using the master method, by applying the change of variable $m = \log n$.

   Note: challenge questions are not only difficult, but also they go beyond the scope of Comp 160. In order to find the solution you may need some insight that is not directly taught in class. We put these kind of questions for students that are interested in going beyond the content of the course.