University of Waterloo
CS 341 — Algorithms
Fall 2013
Assignment 2

Due: Wednesday, October 9, 2013, at 3:00pm.
Late Assignment Accepted Until: Thursday, October 10, 2013, at 3:00pm.

1. Consider the proof of Lemma 4.3 of the Master Theorem provided in the course text, which proves that equation 4.23 is in $O(n^{\log_b a})$.

$$\begin{aligned}
\sum_{j=0}^{\log_b n-1} a^j \left(\frac{n}{b^j}\right)^{\log_b a-\epsilon} &= n^{\log_b a-\epsilon} \sum_{j=0}^{\log_b n-1} \left(\frac{ab^\epsilon}{b^{\log_b a}}\right)^j \\
&= n^{\log_b a-\epsilon} \sum_{j=0}^{\log_b n-1} (b^\epsilon)^j \\
&= n^{\log_b a-\epsilon} \left(\frac{b^{\epsilon \log_b n}-1}{b^\epsilon-1}\right) \\
&= n^{\log_b a-\epsilon} \left(\frac{n^\epsilon-1}{b^\epsilon-1}\right).
\end{aligned}$$

   Expand on each step of the proof: to get from one line of your proof to the next, only a single property or log formula may be used and you should state which one you used. [5 marks]

2. Prove that $\lg n$ is not polynomially bounded from below, i.e. that $\lg n \notin \Theta(n^x)$ for any $x > 0$. Use this fact to show that the following recurrence relation cannot be solved using the master theorem as stated in the course text.[5 marks]

$$T(n) = \begin{cases} 4T\left(\frac{n}{2}\right) + \frac{n^2}{\lg n} & \text{if } n > 1 \\ \Theta(1) & \text{if } n = 1. \end{cases}$$

3. Solve the recurrence relation stated above using the guess-and-verify method.[5 marks]

4. For the proof of the median-of-median method, the input was divided into groups of five. Derive recurrence relations for two variations of this method: one that divides the input into groups of seven and one that divides the input into groups of three. Comment on whether the choice of group size has any effect on the asymptotic run time of the algoritm. [10 marks]

5. Using the recursion-tree method, solve the following recurrences relation, $T(n) = 7T(n/2) + n^2$, if $n > 1$ and $T(1) = 1$. You may assume that n is an even power of 2. Show your work in detail and express your answer as a Θ bound on $T(n)$. [5 marks]

6. Use a recurrence tree to establish a guess and then use the guess-and-verify method to solve the following recurrence: $T(n) = T(n/2) + T(n/4) + T(n/8)$ if $n > 0$ and $T(1) = 1$ for $n = 1$. You may assume n is an even power of 8. [10 marks]

7. Prove that the greedy algorithm for the interval colouring problem discovers the optimal solution.[5 marks]

8. Create an algorithm that sorts an array, A, of size n where each element is a distinct number. The array is approximately sorted, meaning that every element in A is at most k positions away from its position once A is sorted (for some constant k). For convenience, assume that n is an even power of 2. Derive the upper bound on the asymptotic runtime complexity of your algorithm. If possibly, sort this array in $O(n)$ time. [20 marks]

9. You are in the process of developing a sorting algorithm that runs on a computer cluster with m nodes. You break your input (of size nm) into m equally sized arrays and send each one to a node for sorting. Then each node sends their results back to a single node for further processing.

   (a) Divise an effecient algorithm for combining the results from each node into a single sorted array using a divide-and-conquer approach. Assume this algorithm uses only one node. Given a brief argument for the $O(nm)$ bound on the step that combines the m arrays into a single sorted array. [10 marks]

   (b) Now imagine you know of a data structure that given n elements has the following runtime complexities:

   - Finding the minimum element is $\Theta(1)$,
   - Deleting the minimum element is $\Theta(\lg n)$,
   - Inserting an element is $\Theta(\lg n)$.

   Use this data structure to create an algorithm with the same asymptotic upper bound as your divide-and-conquer algorithm. Again, assume this algorithm uses only one node. [10 marks]

   (c) Finally, comment on the conditions when the divide-and-conquer algorithm would be preferred and when the data structure approach would be preferred.[5 marks]

10. Exercise 9.3-6 on page 223 of the course text. [20 marks]

11. Given n boxes each with a single hidden number inside, and a test procedure that decides if two boxes contain the same or different numbers, determine if there is a number which is present in the majority of the boxes, i.e. whether there are more than $n/2$ boxes with the same hidden number in $O(n \log n)$ time. [20 marks]