University of Houston

COSC 3320: Algorithms and Data Structures
Summer 2015

Homework 2

Due June 17, at the start of class

1. When possible, apply the Master Theorem to give asymptotic bounds for $T(n)$ for the following recurrences:

   (a)
   $$T(n) = \begin{cases} 1 & \text{if } n = 1, \\ 3T(n/3) + n/2 & \text{if } n > 1. \end{cases}$$

   (b)
   $$T(n) = \begin{cases} 4 & \text{if } n = 1, \\ 4T(n/2) + 16n^{15/7} & \text{if } n > 1. \end{cases}$$

   (c)
   $$T(n) = \begin{cases} 1 & \text{if } n = 1, \\ T(n/2 + 2) + n^2 & \text{if } n > 1. \end{cases}$$

   (d)
   $$T(n) = \begin{cases} 1 & \text{if } n = 1, \\ 4T(n/2) + n/\log n & \text{if } n > 1. \end{cases}$$

   (e)
   $$T(n) = \begin{cases} 1 & \text{if } n = 1, \\ \log n \cdot T(n/2) + n^2 & \text{if } n > 1. \end{cases}$$

   (f)
   $$T(n) = \begin{cases} 1 & \text{if } n = 1, \\ 2T(n/2) + n/\log n & \text{if } n > 1. \end{cases}$$

2. Consider the following recurrence relation, and assume n to be a power of four.

   $$T(n) = \begin{cases} n & \text{if } n = 1, \\ 2T(n/4) + \sqrt{n} & \text{if } n = 4^d, d > 0. \end{cases}$$

   (a) Apply the Master Theorem to have an asymptotic bound for $T(n)$.
   (b) Determine the exact value of $T(n)$ using the unfolding technique.
   (c) Prove by induction the correctness of the above solution.
   (d) Verify that the above solution is coherent with the asymptotic bound obtained in (a).

3. Consider the following recurrence relation, and assume n to be a power of two.

$$T(n) = \begin{cases} n & \text{if } n \in \{1, 2\}, \\ T(n/2) + T(n/4) + n & \text{if } n = 2^d, d > 1. \end{cases}$$

   (a) Draw the recursion tree for $n = 32$.

   (b) Determine the number ℓ of levels of the recursion tree, and an upper bound to the total cost associated to level i, with $0 \leq i < \ell$.

   (c) From (b), determine an upper bound to $T(n)$, and argue that this bound is asymptotically tight.

4. Let $S = S[0], S[1], \ldots, S[n-1]$ be a sequence of n elements on which a total order relation is defined. Recall that an *inversion* in S is a pair of elements $S[i], S[j]$ such that $S[i] > S[j]$ and $i < j$. Give a recursive algorithm that determines the number of inversions in S in time $O(n \log n)$. (Hint: adapt the Merge-Sort strategy.)