Illinois Institute of Technology
Department of Computer Science

Homework 1 Solutions

CS 430 Introduction to Algorithms
Fall Semester, 2010

1. **Problem 2.3-3 on page 39**

   Solution:

   We will verify by mathematical induction that $T(n) = n \lg n$ is the solution to the recurrence

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

   Basis case: Let $n = 2$, then $T(n) = 2 = 2(1) = 2 \lg 2$.

   Inductive step: We assume the inductive hypothesis that $T(n/2) = n/2 \lg n/2$, where $n/2 = 2^k$ for some $k \geq 1$. Now we need to show that $T(n) = n \lg n$, where $n = 2^{k+1}$.

   $$\begin{aligned} T(n) &= 2T(n/2) + n && (2)\\ &= 2(n/2 \lg n/2) + n && (3)\\ &= n(\lg n - \lg 2) + n && (4)\\ &= n \lg n - n + n && (5)\\ &= n \lg n && (6) \end{aligned}$$

2. **Problem 2.3-4 on page 39**.

   Solution:

   If $n > 1$, then the recurrence is $T(n) = T(n-1) + \Theta(n)$. To sort n elements, we sort the first $(n-1)$ elements using our recursive insertion sort procedure. Since there are $n-1$ elements, this takes $T(n-1)$ time. Then we insert the last element, $A[n]$, into the inserted list. This step takes $\Theta(n)$ time.

   Although, not required by the problem, we can solve this recurrence using the operator method presented in class. To do this, we rewrite the recurrence as

   $$t_n = t_{n-1} + cn, \tag{7}$$

   where c is a positive constant. The annihilator for this recurrence is $(E-1)(E-1)^2 = (E-1)^3$. Thus, the general solution is $t_n = (\alpha + \beta n + \gamma n^2)1^n = \Theta(n^2)$. This shows that the recursive insertions sort takes $T(n) = \Theta(n^2)$ time which is the same as the non-recursive version.

3. **Problem 2-3(a) on page 41**.

   Solution: Since each iteration of the loop takes a constant amount of time (i.e. the time taken for each iteration does not depend on the degree of the polynomial), and the loop executes $\Theta(n)$ times, this code fragment takes $\Theta(n \cdot 1) = \Theta(n)$ time.

4. **Problem 3-3a, fourth column only , on pages 61-62**

   Solution: Rank as follows (from slowest to fastest):

$$\lg(n!), \underline{n^2, 4^{\lg n}}, n^{\lg\lg n}, 2^n$$

   $n^2 = \omega(\lg(n!))$ since $\lg n! \leq \lg n^n = n \lg n$ and $n^2 = \omega(n \lg n)$.
   $4^{\lg n} = \Theta(n^2)$ since $4^{\lg n} = n^{\lg 4} = n^2$.
   $n^{\lg\lg n} = \omega(n^2)$ since $\lg\lg n = \omega(2)$.
   $n^{\lg\lg n} = \omega(2^n)$ since $\lim_{n\to\infty} \dfrac{n^{\lg\lg n}}{2^n} = 0$. This may be easier to see if we compare the log of both functions, then we get $\lg n^{\lg\lg n} = (\lg\lg n)(\lg n) \leq (\lg n)(\lg n) = \lg^2 n$, $\lg 2^n = n$, and $\lim_{n\to\infty} \dfrac{\lg^2 n}{n} = 0$.

5. **Problem 4-3(a) on page 108**

   Solution: If we use a domain transformation then we can use the annihilator method to solve the recurrence $T(n) = 4T(n/3) + n \lg n$. We define the new function $t(k) = T(3^k)$. This gives us the recurrence

$$\begin{aligned} t(k) = T(3^k) &= 4 \cdot T(3^k/3) + 3^k \lg 3^k && (8)\\ &= 4 \cdot T(3^{k-1}) + 3^k \lg 3^k && (9)\\ &= 4 \cdot t(k-1) + 3^k \lg 3^k && (10)\\ &= 4 \cdot t(k-1) + 3^k \frac{1}{\log_3 2} \log_3 3^k && (11)\\ &= 4 \cdot t(k-1) + (\frac{1}{\log_3 2})(k)3^k. && (12) \end{aligned}$$

   The annihilator for the homogeneous part of the recurrence is $(E-4)$ and the annihilator for the nonhomogeneous part is $(E-3)^2$. Thus, the general solution is $t(k) = \alpha 4^k + (\beta + \gamma k)3^k = \Theta(4^k)$. After substituting $n = 3^k$ we get $T(n) = \Theta(4^{\log_3 n}) = \Theta(n^{\log_3 4})$.