CPSC 320: Intermediate Algorithm Design and Analysis
Assignment #6, due Thursday, March 15$^{\text{th}}$, 2012 at 11:00

[3] 1. Derive an asymptotic upper bound for the worst-case running time of a divide-and-conquer algorithm whose recurrence relation is $T(n) \leq 2 \cdot T(\frac{n}{2}) + c \cdot \frac{n}{2}$, where $T(n)$ is defined as in class.

[15] 2. Given a devide-and-conquer algorithm whose recurrence relation is given by $T(n) \leq 4 \cdot T(\frac{n}{4}) + c \cdot n^2$, where n denotes the size of the input problem.

[3] a. Draw the corresponding tree for levels 0, 1 and 2. Next to each node, write the size of the problem corresponding to that node.

[12] b. Derive an upper bound for the worst-case running time of the algorithm. Before you start your derivation of the asymptotic upper bound, first specify

- (a) the number of sub-problems you are dealing with at a given level i,
- (b) the size of *each* sub-problem at a given level i,
- (c) the amount of work required to solve *each* sub-problem at level i,
- (d) the total amount of work required to solve *all* sub-problems at level i,
- (e) the total number of levels required in the algorithm.

Hint: You may want to first remind yourself of our proof in class for obtaining an asymptotic upper bound for the worst-case running time of generic algorithm 3. You may use the following equations:
(1) Geometric sum: $\sum_{i=0}^{m} q^i = \frac{q^{m+1}-1}{q-1} = \frac{1-q^{m+1}}{1-q}$, for $q \neq 1$,
(2) $a^{log_c(b)} = b^{log_c(a)} = c^{log_c(a) \cdot log_c(b)}$,
(3) $log_c(1/x) = -log_c(x)$.

[15] 3. Given a devide-and-conquer algorithm whose recurrence relation is given by $T(n) \leq 4 \cdot T(\frac{n}{4}) + c \cdot n^3$, where n denotes the size of the input problem. Derive an upper bound for the worst-case running time of the algorithm. Before you start your derivation of the asymptotic upper bound, first specify:

- (a) the number of sub-problems you are dealing with at a given level i,
- (b) the size of *each* sub-problem at a given level i,
- (c) the amount of work required to solve *each* sub-problem at level i,
- (d) the total amount of work required to solve *all* sub-problems at level i,
- (e) the total number of levels required in the algorithm.

Hint: In your proof you may use the following:
(1) Geometric sum: $\sum_{i=0}^{m} q^i = \frac{q^{m+1}-1}{q-1} = \frac{1-q^{m+1}}{1-q}$, for $q \neq 1$,

(2) $a^{log_c(b)} = b^{log_c(a)} = c^{log_c(a) \cdot log_c(b)}$,

(3) $log_c(a^b) = b \cdot log_c(a)$.