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

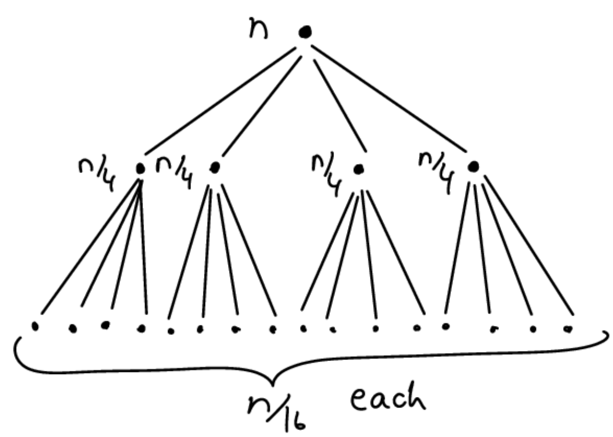
[3] 1. Derive an asymptotic upper bound for the worst-case running time of a divide-and-conquer algorithm whose recurrence relation is $T(n) \leq 2 \cdot T(\frac{n}{2}) + c \cdot \frac{n}{2}$, where $T(n)$ is defined as in class.

Solution: This recurrence relation fits the one underlying generic algorithm 2 introduced in class for $q = 2$ and a new constant $c' = \frac{c}{2}$, i.e. $T(n) \leq q \cdot T(\frac{n}{2}) + c' \cdot n$. Based on the proof in class, we can thus directly conclude that this algorithm requires $\mathcal{O}(n \log(n))$ time.

[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.

Solution: The tree for levels 0, 1 and 2:



[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) $log_c(1/x) = -log_c(x)$.

Solution : We have:

- (a) The number of sub-problems at a given level i is 4^i.
- (b) The size of *each* sub-problem at a given level i is $n/4^i$.
- (c) The amount of work required to solve *each* sub-problem at level i is $c \cdot (n/4^i)^2$.
- (d) The total amount of work required to solve *all* sub-problems at level i is (c) times (a), i.e. $4^i \cdot c \cdot (n/4^i)^2 = cn^2/4^i$.
- (e) The total number of levels required in the algorithm is $log_4(n)$.

We can now use the above quantities in the specified recurrence relation to obtain an asymptotic upper bound.

$$
\begin{aligned}
T(n) &\leq 4 \cdot T(\frac{n}{4}) + c \cdot n^2 \\
&= \sum_{i=1}^{log_4 n} c\frac{n^2}{4^{i-1}} \qquad \text{(summing (d) over all levels)} \\
&= cn^2 \sum_{i=1}^{log_4 n} (\frac{1}{4})^{i-1} \\
&= cn^2 \sum_{i=0}^{log_4 n - 1} (\frac{1}{4})^{i} \qquad \text{(index shift)} \\
&= cn^2 \frac{1-(1/4)^{log_4 n}}{1-(1/4)} \qquad \text{(using (1) geometric sum)} \\
&= cn^2 \frac{1-n^{log_4 (1/4)}}{1-(1/4)} \qquad \text{(using (2))} \\
&= cn^2 \frac{1-n^{-1}}{1-(1/4)} \qquad \text{(using (3))} \\
&= \frac{4}{3}cn^2(1-n^{-1}) \\
&= \frac{4}{3}cn^2 - \frac{4}{3}cn \\
&\leq \frac{4}{3}cn^2
\end{aligned}
$$

Our algorithm therefore requires $\mathcal{O}(n^2)$ time.

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

Solution: We have:

- (a) The number of sub-problems at a given level i is 4^i.
- (b) The size of *each* sub-problem at a given level i is $n/4^i$.
- (c) The amount of work required to solve *each* sub-problem at level i is $c \cdot (n/4^i)^3$.
- (d) The total amount of work required to solve *all* sub-problems at level i is (c) times (a), i.e. $4^i \cdot c \cdot (n/4^i)^3 = cn^3/4^{2i}$.
- (e) The total number of levels required in the algorithm is $log_4(n)$.

We can now use the above quantities in the specified recurrence relation to obtain an asymptotic upper bound.

$$
\begin{aligned}
T(n) &\leq 4 \cdot T(\frac{n}{4}) + c \cdot n^3 \\
&= \sum_{i=1}^{log_4 n} c \frac{n^3}{4^{2(i-1)}} \qquad \text{(summing (d) over all levels)} \\
&= cn^3 \sum_{i=1}^{log_4 n} (\frac{1}{4})^{2(i-1)}
\end{aligned}
$$

$$
\begin{aligned}
&= cn^3 \sum_{i=0}^{log_4 n-1} (\frac{1}{4})^{2i} \qquad \text{(index shift)} \\
&= cn^3 \sum_{i=0}^{log_4 n-1} (\frac{1}{4^2})^{i} \\
&= cn^3 \frac{1-(1/4^2)^{log_4 n}}{1-(1/4^2)} \qquad \text{(using (1) geometric sum)} \\
&= cn^3 \frac{1-n^{log_4(1/4^2)}}{1-(1/4^2)} \qquad \text{(using (2))} \\
&= cn^3 \frac{1-n^{-2}}{1-(1/4^2)} \qquad \text{(using (3))} \\
&= \frac{16}{15} cn^3 (1-n^{-2}) \\
&= \frac{16}{15} cn^3 - \frac{16}{15} cn \\
&\leq \frac{16}{15} cn^3
\end{aligned}
$$

Our algorithm therefore requires $\mathcal{O}(n^3)$ time.