CPSC 320 Midterm 1
Friday, October 13th, 2006

[11] 1. Short Answers

[3] a. Why did we need to use decision trees for proving the $\Omega(n \log n)$ lower bound on the worst-case running time of comparison sorts, instead of an argument based on the algorithm used?

**Solution :** Because we wanted to prove the result for all comparison sorts, and anything we assume about a specific algorithm wouldn't necessarily be true for all of them.

[8] b. Let $\phi : \mathbf{Z}^+ \rightarrow \mathbf{Z}^+$ be defined by

$$\phi(n) = \text{the number of different divisors of } n$$

So, for instance, $\phi(15) = 4$ (the divisors of 15 are 1, 3, 5 and 15) and $\phi(24) = 8$. Mathematicians have proved that $\phi(n) \in O(n^\varepsilon)$ for every $\varepsilon > 0$, and that $\phi(n) \in \omega(\log^k n)$ for every $k \geq 0$.

For each of the following recurrence relations, either state which case of the Master theorem can be used and give a tight bound for $T(n)$ using $\Theta$ notation, or explain briefly why the Master theorem can not be used.

[4] (i) $\begin{cases} 8T(n/4) + n\phi(n) & \text{if } n \geq 2 \\ 1 & \text{if } n = 1 \end{cases}$

**Solution :** First, I must mention a small typo in the question (that did not have any effect on your answer). The statement "that $\phi(n) \in \omega(\log^k n)$ for every $k \geq 0$" is incorrect. Clearly $\phi(n)$ could be as small as 2 (if $n$ is prime). What I should have said was that $\phi(n) \notin O(\log^k n)$ for every $k \geq 0$. That is, you can find integers with more than $\log^k n$ factors, no matter how large $k$ is. Now, on to the actual solution.

Since $n^{\log_b a} = n^{\log_4 8} = n^{1.5}$, and $f(n) \in O(n^{1.25})$ (picking $\varepsilon = 0.25$ for the fact about $\phi(n)$), this is case 1 of the Master theorem. Hence $T(n) \in \Theta(n^{1.5})$.

[4] (ii) $\begin{cases} 2T(n/4) + n\phi(n) & \text{if } n \geq 2 \\ 1 & \text{if } n = 1 \end{cases}$

**Solution :** Since $n^{\log_b a} = n^{\log_4 2} = n^{0.5}$, and $f(n) \in \omega(n)$ the only case of the Master theorem that might be applicable is case 3. We still have to check the regularity condition, however. Because of a missing $\lfloor\rfloor$ in the question, two answers were accepted:

- Without the $\lfloor\rfloor$, and observing that $\phi(n/4) \leq \phi(n)$, and so $2f(n/4) = 2(n/4)\phi(n/4) \leq n\phi(n)/2 < 0.75n\phi(n)$, which means that the regularity condition holds and so $T(n) \in \Theta(n\phi(n))$.

- If you assume that $n/4$ really should have been $\lfloor n/4 \rfloor$, however, then there is no reason to expect that $\phi(n/4)$ isn't much bigger than $\phi(n)$, and so the regularity condition would not hold, which means that we can not apply the Master theorem.

[10] 2. Asymptotic notations

a. Either prove or disprove the following statement:

> For every pair $f$, $g$ of functions from $\mathbf{N}$ to $\mathbf{R}^+$, if for every $n \geq 1$, $f(n) > g(n)$ then $f \notin O(g)$.

**Solution:** This statement is false. Take $f(n) = n + 1$, and $g(n) = n$. Clearly $f(n) > g(n)$ for every $n \geq 1$, and yet $f \in \Theta(g)$.

b. Let $f, g$ be two functions from $\mathbf{N}$ to $\mathbf{R}^+$, and let $m$ and $s$ be the two functions from $\mathbf{N}$ to $\mathbf{R}^+$ defined by $m(n) = \max\{f(n), g(n)\}$ and $s(n) = f(n) + g(n)$. Prove that $m \in \Theta(s)$.

**Solution:** First we show that $m \in O(s)$. Indeed, $m(n) = \max\{f(n), g(n)\} \leq f(n) + g(n)$ and so $m \in O(s)$ using $c = 1$ as our constant.
Now we prove that $m \in \Omega(s)$. Since $f(n) \leq m(n)$, and $g(n) \leq m(n)$, we have $f(n) + g(n) \leq 2m(n)$, and so $s(n) \leq 2m(n)$ which means that $m(n) \geq s(n)/2$. Hence $m \in \Omega(s)$ using $c = 1/2$ as constant.
This proves that $m \in \Theta(s)$.

[9] 3. Prove upper and lower bounds on the function $T(n)$ defined by

$$T(n) = \begin{cases} 2T(n-1) + n2^n & \text{if } n \geq 2 \\ 1 & \text{if } n = 1 \end{cases}$$

Your grade will depend on the quality of the bounds you provide (that is, showing that $T(n) \in \Omega(1)$ and $T(n) \in O(100^n)$, while true, will not give you many marks).

**Solution:** Using a recursion tree, we observe that

- The root does $n2^n$ work.
- The first level contains 2 nodes, each of which handles $n-1$ elements and does $(n-1)2^{n-1}$ work. So the total amount of work done at this level is $2(n-1)2^{n-1} = (n-1)2^n$.
- The next level contains $2^2$ nodes, each of which handles $n-2$ elements and does $(n-2)2^{n-2}$ work. So the total amount of work done at this level is $2^2(n-1)2^{n-2} = (n-2)2^n$.

The tree continues in this way, with level $i$ containing $2^i$ nodes, each of which handles $n-i$ elements and does $(n-i)2^{n-i}$ work. So the total amount of work done at level $i$ is $2^i(n-i)2^{n-i} = (n-i)2^n$.

Therefore

$$\begin{aligned} T(n) &= \sum_{i=0}^{n-1}(n-i)2^n \\ &= 2^n\sum_{i=0}^{n-1}(n-i) \\ &= 2^n\sum_{i=1}^{n} i \\ &= \frac{n(n+1)2^n}{2} \end{aligned}$$

which means that $T(n) \in \Theta(n^2 2^n)$.

[10] 4. Elections for the Mayor position in a town near Crystal Lake (where the story of the horror movie *Friday the 13th* takes place) proceed as follows:

- All of the town's citizens vote for one of the candidates.
- Votes are added up and the candidate with the most votes is found; if he gathered at least $n/4$ votes, where $n$ is the number of voters, then he wins.
- If the candidate with the most votes has fewer than $n/4$ votes, then the candidate with the fewest votes is eliminated and the process starts again.

You have been asked by Jason Voorhees (the current Mayor) to write a *divide and conquer* algorithm that will accomplish the second step of the process. That is, your algorithm will receive an array with $n$ strings (the names of the candidates chosen by each voter), and should return the name(s) of the candidate(s) that have gathered at least $n/4$ votes, along with the number of votes they received. In the case where no candidate received at least $n/4$ votes, your algorithm is allowed not to return any names.

Hint: In every partition of the set of voters into two groups, a candidate with at least one quarter of the votes overall will have received at least one quarter of the votes in one of the two groups.

[8] a. Design a divide and conquer algorithm that returns the desired information. You do not need to use pseudo-code.

**Solution:** Here is the algorithm. To find all candidates with at least $n/4$ votes, given an array $A$ of candidate names, you

i. Divide the array in two equal-sized sub-arrays, and find recursively in each sub-array the up to four candidats that have at least $n/8$ votes.

ii. Then, you sum up the votes obtained by each of the at most $8$ candidates, by going through the array once per candidate.

iii. Finally, you discard any candidates that ended up with less than $n/4$ votes.

When $n \leq 4$, you can solve the problem directly by returning all the candidates in the array.

[2] b. Analyze the running time of the algorithm you described in your answer to part (a).

**Solution :** The algorithm has a running time described by the recurrence

$$T(n) = \begin{cases} 2T(n/2) + \Theta(n) & \text{if } n \geq 5 \\ \Theta(1) & \text{if } n \leq 4 \end{cases}$$

and by case 2 of the Master theorem, it runs in $\Theta(n \log n)$ time.