CPSC 320 Sample Midterm 1
February 2009

[12] 1. Answer each of the questions with either *true* or *false*. You **must** justify each of your answers; an answer without a justification will be worth at most 1.5 out of 4.

[4] a. If we can use the Master theorem to determine the solution to a recurrence relation, then we can also obtain that solution by drawing the corresponding recursion tree.

Solution : This is true: we proved the Master theorem by drawing a recursion tree (Lemma 1), and then evaluating the resulting summation.

[4] b. Let f, g be two functions from $\mathbf{N}$ into $\mathbf{R}^+$. Assuming that $\lim_{n\to\infty} f(n)/g(n)$ exists, we can use its value to determine whether or not f is in $O(g)$.

Solution : This is true: if $\lim_{n\to\infty} f(n)/g(n) = 0$ then $f \in o(g)$, and hence $f \in O(g)$. If $\lim_{n\to\infty} f(n)/g(n)$ is a positive real number, then $f \in \Theta(g)$ and so $f \in O(g)$. Finally if $\lim_{n\to\infty} f(n)/g(n) = +\infty$ then $f \in \omega(g)$, and therefore $f \notin O(g)$.

[4] c. In class, we proved an $\Omega(n \log n)$ lower bound on the worst-case running time of any algorithm that can be used to sort a sequence of n values.

This is false: we only proved an $\Omega(n \log n)$ lower bound on the worst-case running time of any comparison sort. We did not prove that there isn't some other type of algorithm that might be able to sort a sequence of n values faster.

[9] 2. Consider an algorithm `Confusing` whose running time $T(n)$ is described by the recurrence relation $T(n) = 5T(n/8) + g(n)$, with $T(n) \in \Theta(1)$ for $n \leq 7$. Note that $\log_8 5 \approx 0.774$.

[3] (a) Suppose that $g(n) = n$. What is the running time of algorithm `Confusing`? Express your answer using Θ notation (don't forget to justify it).

Solution : Since $g(n) = n$, and $n \in \Omega(n^{\log_8 5+0.2})$, the only possible case would be case 3. Now we need to check the regularity condition: $ag(n/b) = 5g(n/8) = (5/8)n < (3/4)n$, so $\delta = 3/4$ works and the regularity condition holds. Therefore we can conclude that $T(n) \in \Theta(n)$.

[3] (b) Suppose that $g(n) = \sqrt{n}$. What is the running time of algorithm `Confusing`? Express your answer using Θ notation (don't forget to justify it).

Solution : Since $g(n) = n$ and $\sqrt{n} \in O(n^{\log_8 5-0.2})$, this is case 1 of the Master theorem, hence $T(n) \in \Theta(n^{\log_8 5})$.

[3] (c) Describe as precisely as possible the functions $g(n)$ that would make the running time of algorithm `Confusing` be in $\Theta(n^{\log_8 5} \log^2 n)$?

Solution: In order for the worst-case running time of algorithm `Confusing` to be in $\Theta(n^{\log_8 5} \log^2 n)$, $g(n)$ must be in $\Theta(n^{\log_8 5} \log n)$.

[6] 3. Prove or disprove that $3^{n+2} + 5 \in O(3^{n-1})$

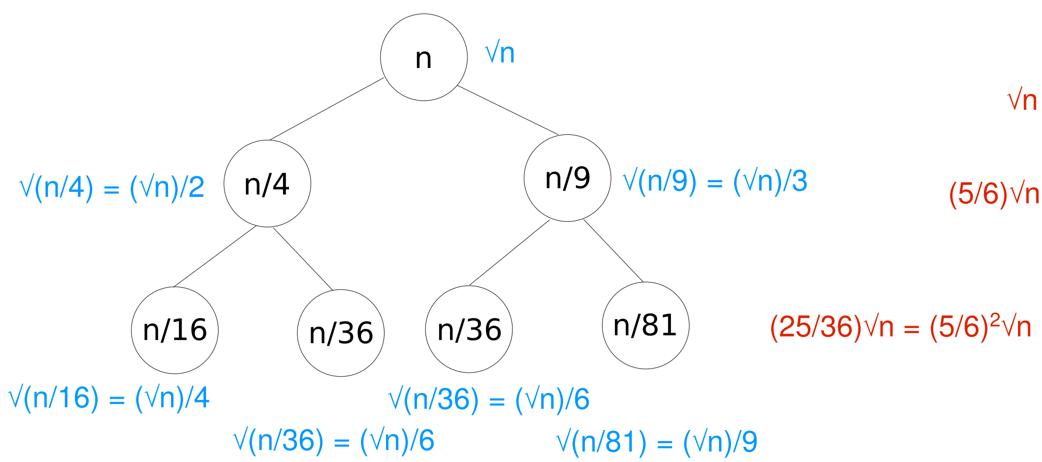
Solution: This is true. Pick $c = 32$ and $n_0 = 1$. Then $3^{n+2} + 5 = 27 \cdot 3^{n-1} + 5 \leq 27 \cdot 3^{n-1} + 5 \cdot 3^{n-1}$ (for $n \geq 1$), and so $3^{n+2} + 5 \leq 32 \cdot 3^{n-1} = c3^{n-1}$.

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

$$T(n) = \begin{cases} T(n/4) + T(n/9) + \Theta(\sqrt{n}) & \text{if } n \geq 9 \\ \Theta(1) & \text{if } n \leq 8 \end{cases}$$

You may ignore floors and ceilings. 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: Let us first establish an upper bound for $T(n)$, by drawing a recursion tree.



As we can see from the recursion tree in the figure, the children of each node N do five-sixth the amount of work done at N. Thus the amount of work done by row i of the recursion tree is at most $(5/6)^i \sqrt{n}$, up to the level where the last leaf occurs. Thus the total amount of work is

$$\sum_{i=0}^{\log_4 n} (5/6)^i \sqrt{n} \leq \sum_{i=0}^{\infty} (5/6)^i \sqrt{n} = \sqrt{n} \sum_{i=0}^{\infty} (5/6)^i = \frac{1}{1 - 5/6} \sqrt{n} = 6\sqrt{n}.$$

This means that $T(n) \in O(\sqrt{n})$.

For the lower bound, observe that the root of the recursion tree performs $\sqrt{n}$ work, and hence $T(n) \in \Omega(\sqrt{n})$. Putting the upper and lower bounds together, we conclude that $T(n) \in \Theta(\sqrt{n})$.

[9] 5. [7] a. Design a divide and conquer algorithm that takes as input an unordered array of elements, and returns the *second largest* element of the array. Hint: your algorithm will actually need to return two values instead of one.

Solution: The idea is to return both the largest and second largest elements of the array from each call.

```
Algorithm SecondLargestPublic(A)
  return SecondLargest(A, 0, length[A] - 1)[1]

Algorithm SecondLargest(A, first, last)
  if (first = last) then
    return (A[first], −∞)
  endif

  mid ← ⌊(first + last)/2⌋
  left ← SecondLargest(A, first, mid)
  right ← SecondLargest(A, mid+1, last)

  return largest and second largest elements of
             (left[0], left[1], right[0], right[1])
```

Note that computing the largest and second largest elements of left[0], left[1], right[0], right[1] is easily done in constant time, since only four elements are involved.

[2] b. Analyze the running time of the algorithm you described in your answer to part (a) by writing a recurrence relation for it, and solving the recurrence using the Master theorem.

Solution: Let $T(n)$ be the worst-case running time of our algorithm. It is defined by the recurrence

$$T(n) = \begin{cases} T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + \Theta(1) & \text{if } n \geq 2 \\ \Theta(1) & \text{if } n = 1 \end{cases}.$$

Because $n^{\log_b a} = n^{\log_2 2} = n$, and $\Theta(1) \in O(n^{1-0.5})$, we are in case 1 of the Master theorem. Therefore $T(n) \in \Theta(n)$.