Homework week 7.

Date:

P-1. What is the Big Oh of this code snippet? Work out the computational complexity of the following piece of code assuming that $n = 2^m$

```
for (int i=n; i>0; i--) {
    for (int j=1; j<n; j*=2) {
        for (int k=0; k<j; k++) {
            ... // constant number c of operations
        }
    }
}
```

Sol-n: The outer for-loop goes around n times. For each i, the next loop goes around $m = \log_2 n$ times, because of doubling the variable j. For each j, the innermost loop by k goes round j times, so that the 2 inner loops together go around $1 + 2 + 4 + \dots + 2^{m-1} = 2^m - 1 \approx n$ times. Loops are nested, so the bounds may be multiplied to give that the algorithm is $O(n^2)$

P-2. Always true or not (i.e. false)?

$\forall$. $f(n) = \Theta(f(n/2))$. $\neq O(2^n)$

S-n: Q. False. counterexample: Let $f(n) = 2^{2n}$. By exercise 3.1-4. $2^{2n} \neq O(2^n)$

P-3 Textbook, problems for chapter 4. 4-4. More recurrence examples.

Give asymptotically tight upper & lower bounds for $T(n)$ in each of the following recurrences. Justify your answer.

a) $T(n) = 5T(n/3) + n \lg n$.

S-n: We have $f(n) = n \lg n$ & $n^{\log_b a} = n^{\log_3 5} \approx n^{1.465}$. Since $n \lg n = O(n^{\log_3 4 - \epsilon})$ for any $0 < \epsilon \leq 0.46$ by case 1 of the master theorem, we have $T(n) = \Theta(n^{\log_3 5})$

b) $T(n) = 3T(n/3) + n/\lg n$. If we were to draw the recursion tree, depth i of the tree would have 3^i nodes. Each node at depth i incurs a cost of $n/(3^i \lg(n/3^i))$, for a total cost at depth i of $n/\lg(n/3^i)$. Using equation (3.19), we have $\lg(n/3^i) = \frac{\log_3(n/3^i)}{\log_3 2} = \frac{\log_3 n - i}{\log_3 2}$. The number of leaves is $3^{\log_3 n} = n$, each contributing $\Theta(1)$, for a total cost of the recursion tree is

$$\Theta(n) + \sum_{i=0}^{\log_3 n - 1} \frac{n}{\lg 3^i} = \Theta(n) + n \log_3 2 \sum_{i=0}^{\log_3 n - 1} \frac{1}{\log_3 n - i}$$

$= \Theta(n) + n\log_3 2 \sum_{i=0}^{\log_3 n-1} 1$

Date:

$= \Theta(n) + n\log_3 2 \cdot H_{\log_3 n-1}$

$= \Theta(n) + n\log_3 2 \cdot \Theta(1+\log_3 n-1)$

$= \Theta(n \lg \lg n)$

c) $T(n) = 8T(n/2) + n^3\sqrt{n}$. We have $f(n) = n^3\sqrt{n} = n^{7/2}$ & $n^{\log_b a} = n^{\log_2 8} = n^3$ since $n^{7/2} = \Omega(n^{3+\epsilon})$ for $\epsilon = 1/2$. We lack at the regularity condition in case 3 of the master theorem we have $af(n/b) = 8(n/2)^3\sqrt{n/2} = n^{7/2}/\sqrt{2} \le cn^{7/2}$ for $1/\sqrt{2} \le c < 1$. Case 3 applies, and we have $T(n) = \Theta(n^3\sqrt{n})$

d) $T(n) = 2T(n/2-2) + n/2$. Subtracting 2 in the argument shouldn't make much difference, & so this recurrence looks like $T(n) = 2T(n/2) + \Theta(n)$, which fall into case 2 of the master theorem with $k=0$. Therefore, we guess that $T(n) = \Theta(n \lg n)$. We'll prove the upper & lower bounds separately.

The upper bound is easy. We assume that $T(n)$ monotonically increases & so $T(n) = 2T(n/2-2) + n/2 \le 2T(n/2) + n/2$. We can use case 2 of the master theorem with $k=0$ for the upper bound, getting $T(n) = O(n \lg n)$.

For the lower bound, we use a substitution proof, which relies on the inequality $n/2 - 2 \ge n/4$ for $n \ge 8$. We assume that $T(n) \ge cn \lg n$ for some positive constant c that we will choose. We have

$$\begin{aligned} T(n) &= 2T(n/2-2) + n/2 \\ &\ge 2c(n/2-2)\lg(n/2-2) + n/2 \\ &= cn\lg(n/2-2) - 4c\lg(n/2-2) + n/2 \\ &\ge cn\lg(n/4) - 4c\lg(n/2) + n/2 \quad \text{for } n \ge 8 \\ &= cn\lg n - 2cn - 4c\lg n + 4c + n/2 \\ &\ge cn\lg n \end{aligned}$$

if $-2cn - 6c\lg n + 4c + n/2 \ge 0$ which is equivalent to $c \le n/(4n + 8\lg n - 4)$. Choosing $c \le 1/10$ satisfies this inequality

e) $T(n) = 2T(n/2) + n/\lg n$. This part is similar to part b & in fact a little simpler. If we were to draw the recursion tree. Depth of i of the tree would have 2^i nodes. Each node at depth i incurs a cost of $n/(2^i \lg(n/2^i))$, for a total cost at depth i of $n/\lg(n/2^i)$. The number of leaves is $2^{\lg n} = n$, each contributing $\Theta(1)$, for a total contribution from the leaves of $\Theta(n)$. Thus the total cost of the recursion tree is

Date:

$$\Theta(n) + \sum_{i=0}^{\lg n - 1} \frac{n}{\lg(n/2^i)} = \Theta(n) + n \sum_{i=0}^{\lg n - 1} \frac{1}{\lg n - i}$$

$$= \Theta(n) + n \sum_{i=0}^{\lg n - 1} \frac{1}{i}$$

$$= \Theta(n) + n \cdot H_{\lg n - 1}$$

$$= \Theta(n) + n \cdot \Theta(\lg \lg n - 1)$$

$$= \Theta(n \lg \lg n)$$

We did a careful accounting in our recursion tree, but we can use this analysis as a guess that $T(n) = \Theta(n \lg \lg n)$. If we were to do a strait substitution proof, it would be rather involved. Instead, we will show by substitution that $T(n) \le n(1 + H_{\lfloor \lg n \rfloor})$ & $T(n) \ge n \cdot H_{\lceil \lg n \rceil}$, where H_k is the kth harmonic number: $H_k = 1/1 + 1/2 + 1/3 + \ldots + 1/k$. We also define $H_0 = 0$. Since $H_k = \Theta(\lg k)$, we have that $H_{\lfloor \lg n \rfloor} = \Theta(\lg \lfloor \lg n \rfloor) = \Theta(\lg \lg n)$ & $H_{\lceil \lg n \rceil} = \Theta(\lg \lceil \lg n \rceil) = \Theta(\lg \lg n)$. Thus, we will have that $T(n) = \Theta(n \lg \lg n)$.

The base case for the proof is for $n = 1$, & we use $T(1) = 1$. Here, $\lg n = 0$ so that $\lg n = \lfloor \lg n \rfloor = \lceil \lg n \rceil$. Since $H_0 = 0$, we have $T(1) = 1 \le 1(1 + H_0)$ & $T(1) = 1 \ge 0 = 1 \cdot H_0$.

For the upper bound of $T(n) \le n(1 + H_{\lfloor \lg n \rfloor})$, we have

$$T(n) = 2T(n/2) + n/\lg n$$

$$\le 2((n/2)(1 + H_{\lfloor \lg(n/2) \rfloor})) + n/\lg n$$

$$= n(1 + H_{\lfloor \lg n - 1 \rfloor}) + n/\lg n$$

$$= n(1 + H_{\lfloor \lg n \rfloor - 1} + 1/\lg n)$$

$$\le n(1 + H_{\lfloor \lg n \rfloor} - 1 + 1/\lfloor \lg n \rfloor)$$

$$= n(1 + H_{\lfloor \lg n \rfloor})$$

where the last line follows from the identity $H_k = H_{k-1} + 1/k$.