EL9343 Homework 2

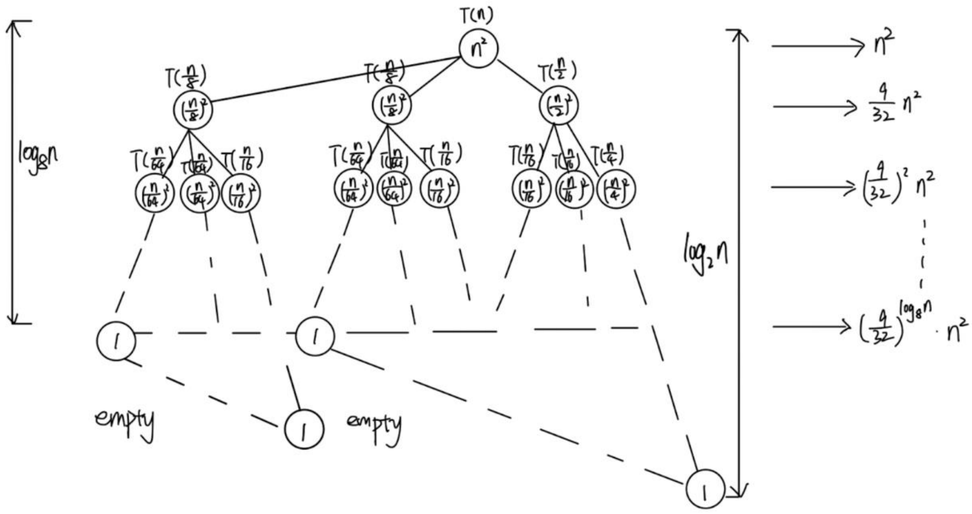
Due: Sept. 21st 8:00 a.m.

1. First use the iteration method to solve the recurrence, draw the recursion tree to analyze.

$$T(n) = T(\frac{n}{2}) + 2T(\frac{n}{8}) + n^2$$

Then use the substitution method to verify your solution.

Part 1 - the iteration method:



The recursion tree is asymmetric and unbalanced. The minimum depth is $\log_8 n$ at the most left side while the maximum is $\log_2 n$ at the most right side. Note that the cost at each depth is reduced by a factor of $\frac{9}{32}$ before reaching the depth $\log_8 n$. In other words, the merging cost at depth k is bounded by $n^2 \times (\frac{9}{32})^k$. Then we can bound $T(n)$ from above and below by,

$$T_{upper}(n) = (1 + (\frac{9}{32}) + (\frac{9}{32})^2 + \cdots + (\frac{9}{32})^{log_2 n})n^2$$

$$T_{lower}(n) = (1 + (\frac{9}{32}) + (\frac{9}{32})^2 + \cdots + (\frac{9}{32})^{log_8 n})n^2$$

$$T_{lower}(n) \leq T(n) \leq T_{upper}(n)$$

Both $T_{upper}(n)$ and $T_{lower}(n)$ are $\Theta(n^2)$, since $\lim\limits_{n\to\infty} \frac{T_{upper}(n)}{n^2} = \lim\limits_{n\to\infty} \frac{T_{lower}(n)}{n^2} = \frac{32}{23}$.
Therefore, $T(n) = \Theta(n^2)$.

Part 2 - substitution method:
The upper bound:
IH: $T(k) \leq dk^2,\ \forall k < n \implies T(n) \leq dn^2$, so we have:

$$T(n) = T(\frac{n}{2}) + 2T(\frac{n}{8}) + n^2 \leq d(\frac{n}{2})^2 + 2d(\frac{n}{8})^2 + n^2 \leq dn^2 \Leftrightarrow d \geq \frac{32}{23}$$

Thus, if we set $d \geq \frac{32}{23}$, then for all n, $T(n) \leq dn^2 \implies T(n) = O(n^2)$.
The lower bound:
IH: $T(k) \geq ck^2,\ \forall k < n \implies T(n) \geq cn^2$, so we have:

$$T(n) = T(\frac{n}{2}) + 2T(\frac{n}{8}) + n^2 \geq c(\frac{n}{2})^2 + 2c(\frac{n}{8})^2 + n^2 \geq cn^2 \Leftrightarrow c \leq \frac{32}{23}$$

Thus, if we set $c \leq \frac{32}{23}$, then for all n, $T(n) \geq cn^2 \implies T(n) = \Omega(n^2)$.
With both upper and lower bounds, we can get $T(n) = \Theta(n^2)$.

2. Use the substitution method to prove that,

$$T(n) = 2T(\frac{n}{2}) + cn\log n$$

is $O(n(\log n)^2)$, where $c > 0$ is a constant. ($\log \equiv \log_2$, in this and the following questions)

IH: $T(k) \le dk(\log k)^2,\ \forall k < n \implies T(n) \le dn(\log n)^2$, so we have:

$$T(n) = T(n) = 2T(\frac{n}{2}) + cn\log n \le 2d\frac{n}{2}(\log\frac{n}{2})^2 + cn\log n \le dn(\log n)^2$$
$$\Leftrightarrow dn(\log n - 1)^2 + cn\log n \le dn(\log n)^2$$
$$\Leftrightarrow d \ge \frac{c}{2 - \frac{1}{\log n}}$$

As $n \to \infty$, $\frac{c}{2-\frac{1}{\log n}}$ monotone decreasing. So for sufficiently large n, we choose $d \ge c$, then we have:

$$d \ge c \ge \frac{c}{2 - \frac{1}{\log 2}} \ge \frac{c}{2 - \frac{1}{\log n}}$$

Therefore, $T(n) = O(n(\log n)^2)$.

3. Solve the recurrence:

$$T(n) = 2T(\sqrt{n}) + (\log\log n)^2$$

(Hint: Making change of variable)

Method 1:
Let $m = \log n$, so $n = 2^m$, and,

$$T(2^m) = 2T(\sqrt{2^m}) + (\log m)^2$$
$$T(2^m) = 2T((2^m)^{\frac{1}{2}}) + (\log m)^2$$
$$T(2^m) = 2T(2^{\frac{m}{2}}) + (\log m)^2$$

Let $S(m) = T(2^m)$, then $S(m) = 2S(\frac{m}{2}) + (\log m)^2$
By master's method, we have $a = b = 2, d = \log_b a = 1, f(m) = (\log m)^2 = O(m^{d-\epsilon})$, for some $\epsilon > 0$, so $S(m) = \Theta(m)$ and $T(n) = \Theta(\log n)$
Method 2:
Let $m = \log\log n$, so $n = 2^{2^m}$, and,

$$T(2^{2^m}) = 2T(\sqrt{2^{2^m}}) + m^2$$
$$T(2^{2^m}) = 2T((2^{2^m})^{\frac{1}{2}}) + m^2$$
$$T(2^{2^m}) = 2T(2^{2^{m-1}}) + m^2$$

Let $S(m) = T(2^{2^m})$, so $S(m) = 2S(m-1) + m^2$, thus we have,

$$S(m) = 2S(m-1) + m^2 = \sum_{i=0}^{m-1} 2^i(m-i)^2$$
$$S(m) = m^2 + 2(m-1)^2 + 4(m-2)^2 + \cdots + 2^{m-1}1^2$$
$$2S(m) = 2m^2 + 4(m-1)^2 + 8(m-2)^2 + \cdots + 2^m 1^2$$
$$\therefore S(m) = 2(2m-1) + 4(2m-3) + 8(2m-5) + \cdots + 2^{m-1}3 + 2^m - m^2$$
$$\text{Similarly, we could have } S(m) = 2^{m+2} + 2^{m+1} - m^2 - 4m - 6 = \Theta(2^m)$$
$$\therefore T(n) = S(m) = \Theta(2^m) = \Theta(\log n)$$

4. You have three algorithms to a problem and you do not know their efficiency, but fortunately, you find the recurrence formulas for each solution, which are shown as follows:

$$\text{A: } T(n) = 5T(\frac{n}{2}) + \Theta(n)$$
$$\text{B: } T(n) = 2T(\frac{9n}{10}) + \Theta(n)$$
$$\text{C: } T(n) = T(\frac{n}{3}) + \Theta(n^2)$$

Please give the running time of each algorithm (In Θ notation), and which of your algorithms is the fastest (You probably can do this without a calculator)?
For A,

$$a = 5, b = 2, f(n) = \Theta(n)$$
$$\therefore d = log_b a = log5 (\approx 2.322), f(n) = O(n^{d-\epsilon}), \text{ for some } \epsilon > 0, \text{ then } T(n) = \Theta(n^d) = \Theta(n^{log5})$$

For B,

$$a = 2, b = \frac{10}{9}, f(n) = \Theta(n)$$
$$\therefore d = log_b a = log_{\frac{10}{9}} 2 (\approx 6.578), f(n) = O(n^{d-\epsilon}), \text{ for some } \epsilon > 0, \text{ then } T(n) = \Theta(n^d) = \Theta(n^{log_{\frac{10}{9}} 2})$$

For C,

$$a = 1, b = 3, f(n) = \Theta(n^2)$$
$$\therefore d = log_b a = log_3 1 = 0, f(n) = \Omega(n^{d+\epsilon}), \text{ for some } \epsilon > 0$$
$$\text{Also, } f(n) = \Theta(n^2) \implies \exists c < 1, \forall \text{large } n, af(n/b) \leq cf(n),$$
$$\text{Then, } T(n) = \Theta(f(n)) = \Theta(n^2)$$

Obviously, $2 < \log_5 < log_{\frac{10}{9}} 2$, so C is the fastest.
Special notes:
For algorithm C, you will get full points if you have written something similar to the above answer. However, there is a flaw when applying master's method: for $f(n) = \Theta(n^2)$, such c may not exist. Let's consider the following case:

$$f(n) = \begin{cases} 0.5n^2 & n = 3k, k \in \mathbb{Z} \\ 10n^2 & \text{otherwise} \end{cases}$$

It's clear that such $f(n) = \Theta(n^2)$, but for large $n = 3k$, where k is integer but not a multiple of 3, $f(\frac{n}{3}) > f(n)$. Thus, master's method couldn't apply here. Instead of master's method, we could use iteration method to solve it, shown as follows,

$$f(n) = \Theta(n^2) \Leftrightarrow \exists c_1, c_2 > 0, c_1 n^2 \leq f(n) \leq c_2 n^2$$
$$T(n) = T(\frac{n}{3}) + f(n^2) = \sum_{i=0}^{\log_3 n} f(\frac{n}{3^i})$$
$$\therefore T(n) \geq c_1 n^2 \sum_{i=0}^{\log_3 n} \frac{1}{9^i} = \Omega(n^2)$$
$$T(n) \leq c_2 n^2 \sum_{i=0}^{\log_3 n} \frac{1}{9^i} = O(n^2)$$
$$\therefore T(n) = \Theta(n^2)$$

$T(n)$ is still $\Theta(n^2)$, even if it doesn't satisfy the requirement of case 3 in master's method. We can see that the requirement of case 3 is a sufficient condition to the statement, but is not a necessary condition. Also, iteration method and substitution method are still the primary ways to solve the recurrence.