CMPUT 275 — Tangible Computing II — Winter 2018
Master Theorem - Solutions

The "Master Theorem" is a "cookbook result" that helps us solving recurrences of the form

$$T(n) = aT(n/b) + O(n^c)$$

for some constants $a > 0$, $b > 1$, $k \geq 0$ and $c \geq 0$. In particular, the theorem tells us that if the above is satisfied then

$$T(n) = \begin{cases} O(n^c)\,, & \text{if } c > \log_b a\,; \\ O(n^{\log_b a})\,, & \text{if } c < \log_b a\,. \end{cases}$$

Furthermore, if $c = \log_b a$ and $f(n) = n^c \cdot \log^k n$ for some $k \geq 0$, then $T(n) = O(n^c \cdot \log^{k+1} n)$.

One can even replace n/b with $\lceil n/b \rceil$ in the recursive call and still have these bounds. Also, we can replace $O()$ by $\Theta()$ throughout (i.e. if $f(n) = \Theta(n^c)$ and $c < \log_b a$ then $T(n) = \Theta(n^{\log_b a})$.

Use the Master Theorem to find the asymptotic growth of T in the following recurrences:

1. $T(n) = 2T(n/4) + 1$

   $a = 2, b = 4, c = 0$
   So $c < \log_2 a$ and we have $T(n) = O(n^{\log_b a}) = O(n^{1/2}) = O(\sqrt{n})$.

2. $T(n) = 2T(n/4) + \sqrt{n}$

   $a = 2, b = 4, c = 1/2$
   So $c = \log_b a$. Note $k = 0$ when we say $f(n) = n^c \cdot \log^k n$ in this case.
   We have $T(n) = O(n^c \cdot \log^{k+1} n) = O(\sqrt{n} \log n)$.

3. $T(n) = 2T(n/4) + n$

   $a = 2, b = 4, c = 1$
   So $c > \log_b a$ and we can say $T(n) = O(n^c) = O(n)$.

4. $T(n) = 2T(n/4) + \sqrt{n} \cdot \log^3 n$

   This is like the second case, except $k = 3$ so $T(n) = O(\sqrt{n} \cdot \log^4 n)$.

5. An optional, more difficult problem which requires some ideas beyond the basic Master Theorem:
   $T(n) = 2T(\sqrt{n}) + \log_2 n$

   Note there are 2^i calls to depth i of the recursion, each with an input of size $n^{(1/2)^i}$. What is the maximum depth? For $i = \log_2 \log_2 n$ we have $n^{(1/2)^i} = 2$, so it suffices to bound the amount of work until depth $i = \lceil \log_2 \log_2 n \rceil$.

   Let $\alpha = \lceil \log_2 \log_2 n \rceil$. The amount of time

   $$\sum_{i=0}^{\alpha} 2^i \cdot \log_2(n^{(1/2)^i}) = 2^i \cdot \sum_{i=0}^{\alpha} 2^i \frac{1}{2^i} \cdot \log_2 n = \log_2 n \sum_{i=0}^{\alpha} 1 = \log_2 n \cdot (\alpha + 1).$$

   So $T(n) = O(\log n \cdot \alpha) = O(\log n \cdot \log \log n)$.