CMPUT 275 — Tangible Computing II Winter 2018
Master Theorem

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

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

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

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

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