How to Solve Recurrences

1. Master Method *
2. Guess & Check

Assumptions:
- all subproblems have same size
- $T(n) \leq \text{constant}$ for $n < n_0$ (where $n_0$ is some small integer)
- $n > n_0$:
  \[
  T(n) = a \cdot T\left(\frac{n}{b}\right) + O(n^d)
  \]
  (constants $a, b, d$ independent of $n$)

Q: What are $a$, $b$, $d$ in words?

- $a$: # of recursive calls
- $b$: factor by which problem shrinks in recursive call
- $d$: characterizes extra work outside recursive call (in combine step)

\[
T(n) = \begin{cases} 
O(n^{d \log_b a}) \\
O(n^d) \\
O(n^{d \log_b a})
\end{cases}
\]

Three Cases
\[
\begin{align*}
a &= b^d \\
a < b^d \\
a > b^d
\end{align*}
\]
Examples (divide into groups & have each group present)

1. Binary Search
   \[ a = 1 \quad \text{(only 1 recursive call)} \]
   \[ b = 2 \quad \text{(new problem is \( \frac{1}{2} \) size of old)} \]
   \[ d = 0 \quad \text{(constant work outside recursive call)} \]
   \[ T(1) = \text{constant} \]
   \[ b^d = 2^0 = 1 = a \quad \Rightarrow \quad \text{case 1} \]
   \[ T(n) = n^d \log n = n^0 \log n = \log n \quad \checkmark \text{Sanity Check!} \]

2. Merge Sort
   \[ a = 2 \quad \text{(2 recursive calls)} \]
   \[ b = 2 \quad \text{(new problems \( \frac{1}{2} \) size of old)} \]
   \[ d = 1 \quad \text{(} O(n) \text{ work outside of recursive call)} \]
   \[ T(1) = \text{constant} \]
   \[ b^d = 2^1 = a \quad \Rightarrow \quad \text{case 1} \quad \checkmark \text{Sanity Check!} \]
   \[ T(n) = n^d \log n = n \log n \]