**Tree Method**

Way to solve certain recurrences

\[
T(n) = a \cdot T\left(\frac{n}{b}\right) + O(n^d) \quad \text{for } n < n^* \quad a, b, d \text{ don't depend on } n
\]

Q: If \( T(n) \) is runtime of an algorithm,

What are \( a, b, d \) in words?

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

Let's Add Up All Work

\[
\text{Input size } n \quad \text{Problem size } n \quad \text{Input size } n
\]

ex: 
- \( a = 3 \)
- \( b = 5 \)
- \( d = 4 \)

\[
A) O\left(\frac{n}{3}\right) \quad B) O\left(\frac{n}{2}\right) \quad C) O\left(\frac{n}{5}\right)^4 \quad D) O\left(\frac{n}{3}\right)^4
\]

What goes here?
Let's Add Up All Work

\[ n \leftarrow \text{Problem size} \]

Ex:

\[ a = 3 \]
\[ b = 5 \]
\[ d = 4 \]

\[ O\left(n^d\right) \]
\[ O\left(n^\left\lceil \frac{\log_b(n)}{d} \right\rceil\right) \]
\[ \frac{n}{5} \]
\[ \frac{n}{25} \]

Input size

\[ n \]
Proof of Tree Method

0. What is $F$ (in terms of $a, b, d$)?

A) $O(\log_b n)$  B) $O(\log d n)$  C) $O(n^{\log_b d})$  D) $O(b^{\log_d n})$

Because at each level, problem size is divided by $b$. $\log_b n$ is number of times $n$ can be divided by $b$ before reaching a constant.

$$\underbrace{\text{constant} \cdot b \cdot b \cdots b = n}$$

$$\frac{F}{c} = n$$

$$b^F = \frac{n}{c}$$

take $\log_b$ on both sides

$$\log_b b^F = \log_b \left( \frac{n}{c} \right)$$

$$F = \log_b n - \log_b c$$

$$F = \log_b n + \text{constant}$$
Q. What is the work done just at level $K$, not at other levels?

- $a^K$ subproblems at level $K$.
- Level $K$ subproblem size: $\frac{n}{b^K}$
- Work outside of recursive call required to solve 1 subproblem

\[ \Rightarrow \text{Total work} \quad a^K \left( \frac{n}{b^K} \right)^d = \left( \frac{a}{b^d} \right)^K n^d \]

Now we add up work done at all levels:

\[ T(n) = n^d \left[ \sum_{K=0}^{\log_b n} \left( \frac{a}{b^d} \right)^K \right] \]
Geometric Series:

$$\sum_{k=0}^{F} r^k = \begin{cases} 
F+1 & \text{if } r = 1 \\
\frac{1-r^{F+1}}{1-r} & \text{otherwise}
\end{cases}$$
PSet:

\[ T(n) = \begin{cases} 
  O(n^d \log n) & \text{if } a = b^d \\
  O(n^d) & \text{if } a < b^d \\
  O(n^{\log_b a}) & \text{if } a > b^d 
\end{cases} \]

This is usually called "master method" or "master theorem".

Master has pretty unpleasant connotations. Also it is not descriptive.

My term: "Tree method"