**The Master Method:**

Do you have a recurrence of the form ? Then this is the method for you! Assume that n/b can be either rounded up or rounded down. Then, one of the following expressions have to be true:

This shit made absolutely no sense in the book, so we gonna follow a YT video instead:

//teta(n^d) = f(n)

<https://www.youtube.com/watch?v=T68vN1FNY4o>

Let’s solve some related tasks:





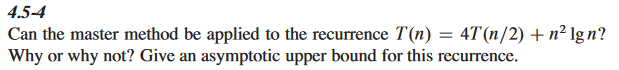


**Recursion trees:**

The act of using tree structures to visualize recurrence. When making such a tree, it is important make sure that each node represents the cost of a single sub-problem. Each parent node’s cost should be the sum of all it’s subnodes’ cost.

Remember that bullshit from tasks above? Turns out, a is the multiplier for how many sub nodes a node has, while is the execution time for each sub node.

**Exercise 3:**



I don’t think the master method can applied, because we technically have 2 functions of n. ().