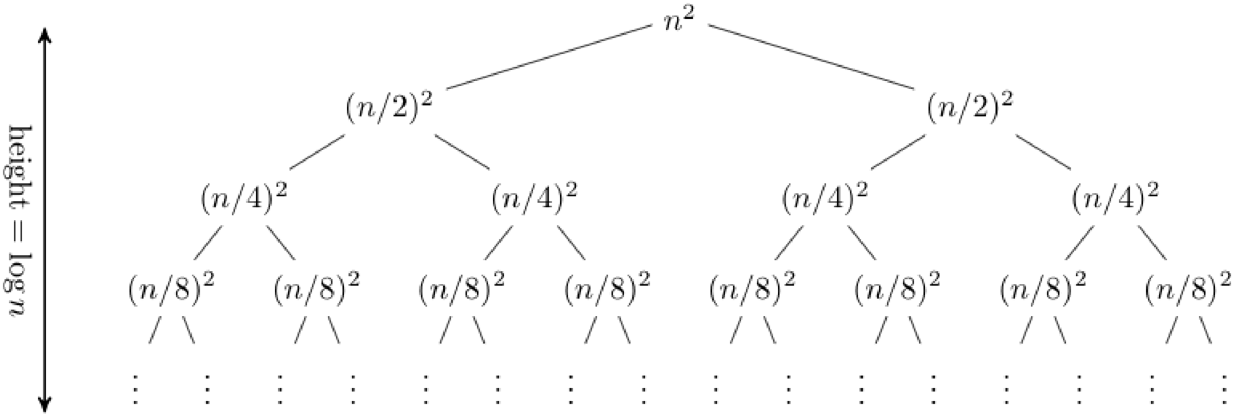
A *recursion tree* is useful for visualizing what happens when a recurrence is iterated. It diagrams the tree of recursive calls and the amount of work done at each call.

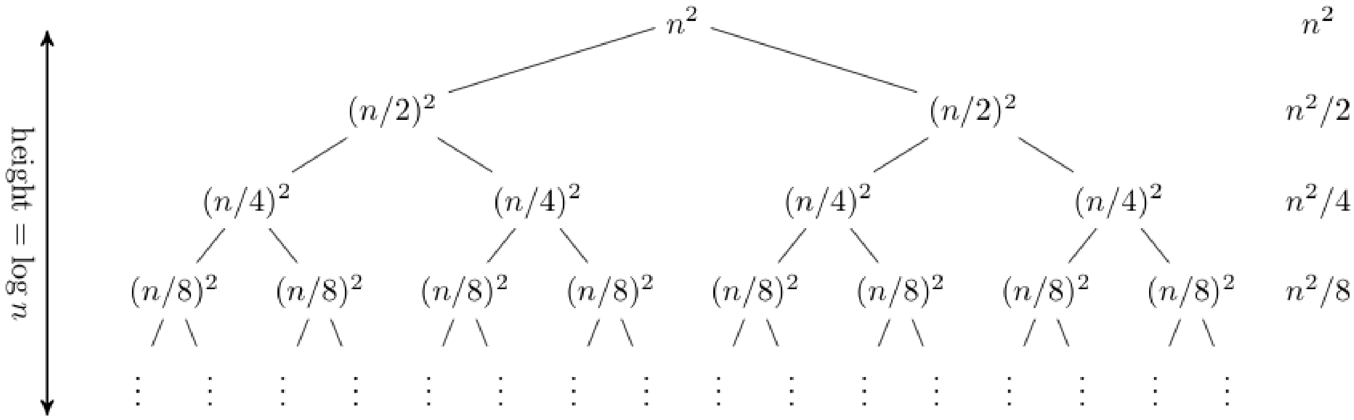
For instance, consider the recurrence

$T(n) = 2T(n/2) + n^2$.

The recursion tree for this recurrence has the following form:



In this case, it is straightforward to sum across each row of the tree to obtain the total work done at a given level:



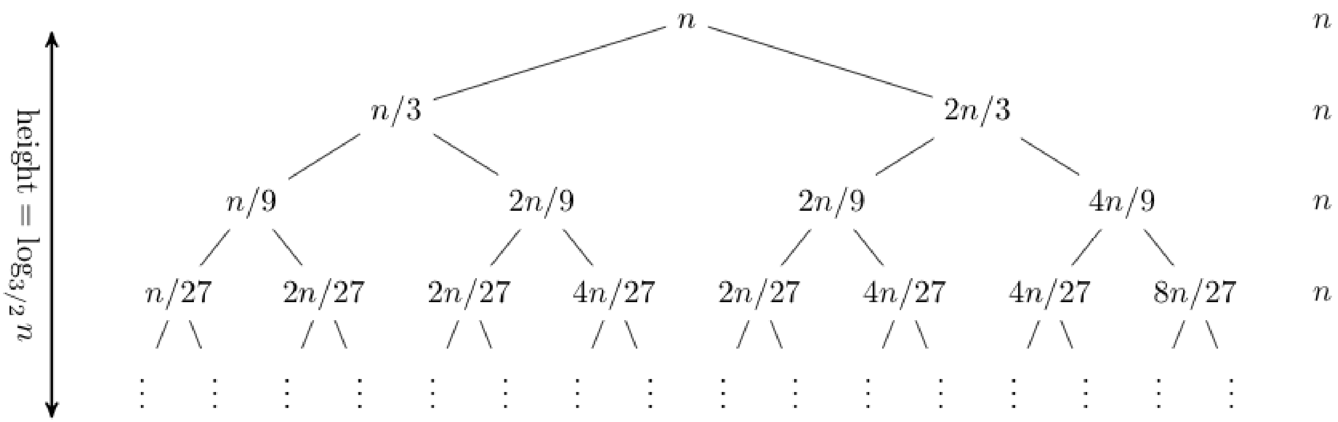
This a geometric series, thus in the limit the sum is $O(n^2)$. The depth of the tree in this case does not really matter; the amount of work at each level is decreasing so quickly that the total is only a constant factor more than the root.

Recursion trees can be useful for gaining intuition about the closed form of a recurrence, but they are not a proof (and in fact it is easy to get the wrong answer with a recursion tree, as is the case with any method that includes "..." kinds of reasoning). As we saw last time, a good way of establishing a closed form for a recurrence is to make an educated guess and then prove by induction that your guess is indeed a solution. Recurrence trees can be a good method of guessing.

Let's consider another example,

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

Expanding out the first few levels, the recurrence tree is:



Note that the tree here is not balanced: the longest path is the rightmost one, and its length is *$\log_{3/2} n$*. Hence our guess for the closed form of this recurrence is *$O(n \log n)$*.