Wednesday 7/23: Recursion Trees

How to build a recursion tree

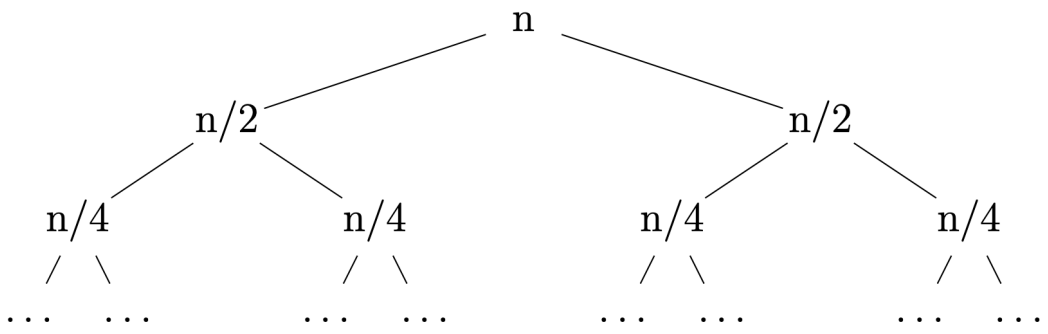
Recursion trees are trees that denote work done at each level, so that we can visualize the behavior of certain recursive definitions for better understanding and calculation.

In order to draw a recursion tree, we can start by building a tree that represents **the input sizes at each level**. We will refer to it as a *size tree* in the following, but note that this is not a formally defined term.

For example, consider the following recursive definition:

$$S(1) = c$$
$$S(n) = 2S(n/2) + n, \forall n \geq 2$$

Each recursive level contains two recursive calls $S(n/2)$, and each recursive call reduces the input size in half. Therefore, a size tree may look like this:

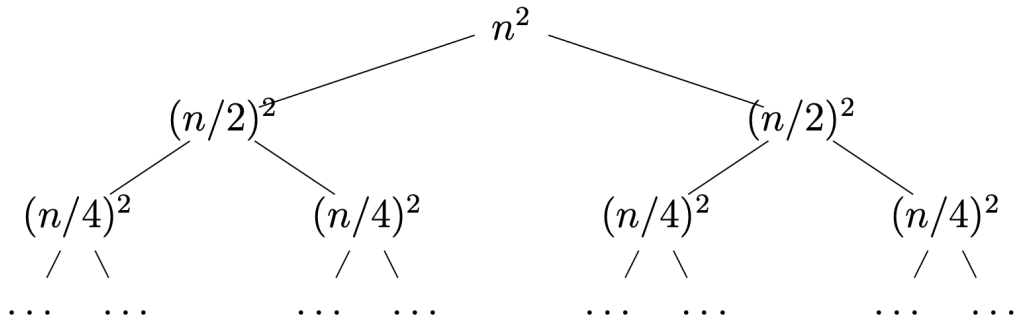


Now let's try to build a recursion tree. At each node, the amount of work contains everything in the recursive formula except the recursive calls to S. Therefore, for $S(n) = 2S(n/2) + n$, we should consider $f(n) = n$ as the amount of work at each node. For example, at the $S(n) = 2S(n/2) + n$ level, the amount of non-recursive work for each node is n; at the $S(n/4) = 2S(n/8) + n/8$ level, the amount of non-recursive work for each node is $n/8$. Also make sure that you plug in the base case formula when you reach the leaves, i.e. $S(1) = c$.

Then, if you draw out a recursion tree for this recursion, you will get the same tree as the size tree. However, this does not always happen. Let's take a look at another recursive definition:

$$P(1) = c$$
$$P(n) = 2P(n/2) + n^2, \forall n \geq 2$$

Similar to $S(n)$, $P(n)$ also includes two recursive calls, and each recursive call reduces the problem size in half. Therefore, $P(n)$ has the same size tree as $S(n)$ (the above tree). However, if we look at the recursion tree, we should note that the amount of work at each node is $g(n) = n^2$. Therefore, the recursion tree should look like the following:



Summing up a recursion tree

One major purpose of building recursion trees is to help us calculate the total amount of work for the entire recursion. To do so, we have a few questions to ask:

- How tall is this tree, i.e. how many levels do we need to expand before we hit the base case $n = 1$?
- For each internal level of the tree, what is the sum of the values in all nodes at that level?
- How many leaf nodes are there? What's the total work for all leaves?

Let's take the recursion tree of $S(n)$ (the first figure) for example.

Because the recursive call reduces its size in half each time ($S(n) = 2S(n/2) + n$), it should take $\log_2 n$ recursive calls to reach the base case $n = 1$ (note that $2^{\log_2 n} = n$). Therefore, this tree has height $\log_2 n$.

At level 0, we have 1 node with work n. At level 1, we have 2 nodes with work $n/2$ each, which sums up to be n. At level 2, we have 4 nodes with work $n/4$ each, which also sums up to be n. Therefore, we have seen a pattern that the total work of each level should sum up to n, and total work for all internal levels sums up to $n * \log_2 n$.

When node size is $n/1$, we have 1 node; when node size is $n/2$, we have 2 nodes; when node size is $n/4$, we have 4 nodes. Therefore, we can infer that when node size is $1 = n/n$, we should have n nodes, i.e. we have n leaves. Each leaf contributes c to the sum, so the total work for all leaves should sum up to cn.

Adding up the work of all internal nodes and leaves, we get $n \log_2 n + cn$ as the total work for this tree.

You should practice calculating the total work for $P(n)$ and check your answer with the solution in Chapter 13.7 of the textbook.