Pre-lecture exercises will not be collected for credit. However, you will get more out of each lecture if you do them, and they will be referenced during lecture. We recommend **writing out** your answers to pre-lecture exercises before class. Pre-lecture exercises usually should not take you more than 20 minutes.

In this pre-lecture exercise, you’ll explore three *recurrence relations*. For each of the following expressions, try to figure out a closed-form expression for \( T(n) \), **when \( n \) is a power of 2**. (Don’t worry about when \( n \) isn’t a power of 2 for now).

If you are feeling stuck, we’ve done the first one for you in two different ways on the next page to give you some inspiration for how to attack the second and the third.

1. \[
    T(n) = \begin{cases} 
        2 \cdot T(n/2) + n & n = 2^i, i > 0 \\
        T(n) = 1 & n = 1
    \end{cases}
\]

2. \[
    T(n) = \begin{cases} 
        T(n/2) + n & n = 2^i, i > 0 \\
        T(n) = 1 & n = 1
    \end{cases}
\]

3. \[
    T(n) = \begin{cases} 
        4 \cdot T(n/2) + n & n = 2^i, i > 0 \\
        T(n) = 1 & n = 1
    \end{cases}
\]
SPOILER ALERT: Here are two solutions to Exercise 1, which you can look at to help you figure out how to do Exercises 2 and 3.

SOLUTION 1. We do as we did with MERGESORT, and imagine a tree with \( \log(n) + 1 \) levels. The top node is labeled “\( n \)”, its two children are labeled “\( n/2 \)”, and so on.

Consider \( T(n) = T(n/2) + T(n/2) + n \). In the context of the tree above, that means that \( T(n) = n + \) (stuff contributed by things in the tree lower than the root). That is,

\[
T(n) = (\text{label on the root}) + (\text{stuff contributed by things lower than the root}).
\]

We can repeat this logic recursively to figure out what that second term is, all the way down to the bottom of the tree, where we have \( T(1) = 1 \). We conclude that \( T(n) \) is equal to the sum, over all the nodes, of the labels on the nodes.

If the root is level 0, then at level \( j \leq \log(n) \), there are \( 2^j \) nodes, each which have label \( n/2^j \). So

\[
\sum_{j=0}^{\log(n)} 2^j \cdot \frac{n}{2^j} = n(\log(n) + 1).
\]

is our answer.

SOLUTION 2. We can do the exact same calculation without the tree, by repeatedly applying our formula.

\[
T(n) = 2T(n/2) + n \\
= 2 (2T(n/4) + n/2) + n \\
= 4T(n/4) + 2n \\
= 4(2T(n/8) + n/4) + 2n \\
= 8T(n/8) + 3n
\]

and at this point we can spot the pattern: for all \( j \leq \log(n) \),

\[
T(n) = 2^j T(n/2^j) + jn.
\]

In order to formally prove that this is true, we should use a proof by induction; that’s called the substitution method and we’ll talk about it on Wednesday. But for now you can convince yourself that this is true.

Once we have this, we can just plug in \( j = \log(n) \), and get

\[
T(n) = 2^{\log(n)} T(n/2^{\log(n)}) + n \log(n) = n \cdot T(1) + n \log(n) = n(\log(n) + 1),
\]

just as before.

\(^1\)Notice that this is a special consequence of the fact that the term we are adding on is exactly \( n \); if it were, say \( 11 \cdot n \), we’d have to multiply all the labels by \( 11 \) before counting their contribution. Or if it were \( \sqrt{n} \), we’d have to take the square root, and so on.