CSE 332 Summer 18 Unrolling

There are two common methods of getting exact closed forms of recurrences (i.e. exact non-recursive formulas for a function initially described as a recurrence). We discussed the tree method in-depth in lecture and section.

1 Unrolling

There is a second common technique called *unrolling*. The fundamental ideas behind unrolling and the tree method are the same – both try to:

1. Find the non-recursive work at the first few levels.
2. Generalize the pattern to describe the work at any level
3. Write a summation
4. Simplify

In our tree method examples, we've done step 2 rather methodically, breaking it down into smaller pieces to make sure we get it right. This makes the tree method less error-prone. But it can also make it quite slow.

Unrolling is an alternative where we try to find the pattern more quickly, without drawing the tree.

Let's go through an example recurrence:

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ 2T(n/2) + n^2 & \text{otherwise} \end{cases}$$

When unfolding, (just like in the tree method) we assume n is very large, and see what the work of the first few levels of recursive calls would be.

$$\begin{aligned} T(n) &= 2T(n/2) + n^2 & \\ &= 2[2T(n/2^2) + (n/2)^2] + n^2 & \text{apply recursive definition on } T(n/2) \\ &= 2^2T(n/2^2) + n^2/2 + n^2 & \text{simplify} \\ &= 2^2[2T(n/2^3) + (n/2^2)^2] + n^2/2 + n^2 & \text{apply recursive definition on } T(n/2^2) \\ &= 2^3T(n/2^3) + n^2/2^2 + n^2/2 + n^2 & \text{simplify} \end{aligned}$$

Now, we stare at the equations and look for a pattern. First, how does the coefficient on $T()$ relate to the the size of the input? Well when the input is $n/2^i$, the coefficient on $T()$ is 2^i. What about the accumulated non-recursive work? It looks like each term is of the form $n/2^j$, where j increases by 1 each time, and the largest j is one less than i.

That gives us the following formula in general: $T(n) = 2^i T(n/2^i) + \sum_{j=0}^{i-1} \frac{n^2}{2^j}$.

We should sanity check this by making sure each of the three equations we've already written meet this formula. An even better sanity check is to go one more level (after we think we've found the pattern) and make sure it still works. In this case, it does!

Our goal now is to get rid of the recursion. To get rid of $T(n/2^i)$, we should cause $n/2^i$ to hit a base case. I.e. we should set $i = \log_2(n)$ so that we get $T(1)$.

$$\begin{aligned}
T(n) &= 2^i T(n/2^i) + \sum_{j=0}^{i-1} \frac{n^2}{2^j} \\
&= 2^{\log_2(n)} T(1) + \sum_{j=0}^{\log_2(n)-1} \frac{n^2}{2^j} \\
&= 2^{\log_2(n)} \cdot 1 + \sum_{j=0}^{\log_2(n)-1} \frac{n^2}{2^j}
\end{aligned}$$

At this point, we just need to simplify (factor out the n^2 and apply the geometric series formula on the second term, and apply log rules on the first term) to get:

$$\begin{aligned}
T(n) &= 2^{\log_2(n)} \cdot 1 + \sum_{j=0}^{\log_2(n)-1} \frac{n^2}{2^j} \\
&= n \cdot 1 + n^2 \frac{(1/2)^{\log_2(n)} - 1}{1/2 - 1} \\
&= n + n^2 \left(1/2 - (1/2)^{\log_2(n)-1}\right) \\
&= n + n^2 \left(1/2 - 2^{1-\log_2(n)}\right) \\
&= n + n^2 \left(1/2 - 2/n\right) \\
&= n + n^2/2 - 2n \\
&= n^2/2 - n
\end{aligned}$$

2 Comparison to Tree Method

Let's try the solving the same recurrence with the tree method.

1. The input size at level i is $n/2^i$ (because we divide by 2 at each level).
2. The number of nodes at level i is 2^i (because each node has 2 children).
3. The work done at level i is: $2^i \cdot \left(\frac{n}{2^i}\right)^2 = 2^i \frac{n^2}{2^{2i}} = \frac{n^2}{2^i}$.
4. The last level of the tree is when $n/2^i = 1$, so level $i = \log_2(n)$.
5. The work done at the base case is $2^{\log_2(n)} \cdot 1 = n$.
6. Summing over all levels we get: $n + \sum_{j=0}^{\log_2(n)-1} \frac{n^2}{2^i}$.
7. Simplifying is the same as it was for unrolling, so we won't write it again.

Good news – we got the same formula each time. In fact, the logic we were using was mostly the same, it's just rearranged a little bit.

If we look at each step of the tree method, we can find most of the same logic in unrolling:

We did steps 1 and 2 when we said this: "First, how does the coefficient on $T()$ relate to the the size of the input? Well when the input is $n/2^i$, the coefficient on $T()$ is 2^i."

We did steps 3 and 6 when we said: "What about the accumulated non-recursive work? It looks like each term is of the form $n/2^j$, where j increases by 1 each time, and the largest j is one less than i. "

We did step 4 by saying: "Our goal now is to get rid of the recursion. To get rid of $T(n/2^i)$, we should cause $n/2^i$ to hit a base case. I.e. we should set $i = \log_2(n)$ so that we get $T(1)$." And step 5 happened when we plugged that decision into the formula.

Step 7 is identical algebra in each.

So why did we spend so much time on the tree method? The difficulty with unrolling is in staring at an equation and generalizing the pattern. In the tree method, we explicitly ask for two pieces: the number of nodes and the work per node, and combine them immediately into a general formula. When unrolling, we're just generating the first few terms of the sequence, and then asserting a pattern. The difficulty is twofold. First, if we simplify too much or too little, we may obscure the pattern, and make our lives much more difficult. Second, even if we think we see the pattern, we may be identifying it incorrectly, for example if there's more than one reasonable way to extend beyond the first few terms. Or even worse, a small algebra mistake can be very hard to notice, as it could lead to some other (still reasonable) pattern. And you don't have the first few levels of the tree to compare to, so you're not likely to catch your mistake.

The benefit of unrolling is that with practice you can do it more quickly. With even more practice you can often see the pattern emerge in the first few terms and reduce even further the chances of finding the wrong pattern.

On Exercise 4 we specifically ask you to use the tree method. After that exercise (including on the midterm) we'll allow you to use whichever method you prefer.