Recitation 9: Recurrence Relations, Continued

Midterm TA reviews. Please take take a few minutes to fill out TA FCQs! They're very helpful for us TAs. You can find the link in your email under the subject line "Computer Science Midterm TA FCQ's."

This week we will be studying methods for finding the asymptotics of the runtime of an algorithm. The way we do this is by writing down a ***recurrence relation*** which relates a larger instance of a problem to a smaller instance of that problem. We can then unroll this recurrence relation down to its base case, and count the amount of unrolling that we need to do. Sometimes unrolling won't work, and we need to use the Tree Method instead.

0 Tree Method

You may have noticed that the recurrences we solved by unrolling were of the form

$$T(n) = aT(n-b) + cn^d$$

and each step can be unrolled from the previous one directly. Sometimes this is not the case, especially in recurrences of the form

$$T(n) = aT\left(\frac{n}{b}\right) + cn^d$$

These recurrences can be more easily solved by drawing a tree. The root of this tree corresponds to $T(n)$, and it branches into a children, each corresponding to an instance $T(n/b)$. A cost of cn^d is added to each node.

1

Try to solve the following recurrences using the tree method:

a. Draw the first few layers of the tree

b. How many iterations does it take to reach the base case?

c. Sum the work over all the nodes

d. Simplify the result

1.1

$$T(n) = \begin{cases} 3T(n/6) + n^2 & n > 3 \\ 5 & n \leq 3 \end{cases}$$

1.2

$$T(n) = \begin{cases} 2T(n/5) + \log_5 n & n > 1 \\ 2 & n \leq 1 \end{cases}$$

2 3-Mergesort

Let's consider a new algorithm for sorting numbers. It's a lot like Mergesort, but instead of splitting the list in half at each step, we split the list into thirds.

a. Write pseudocode for 3-Mergesort

b. Write down the recurrence relation for 3-Mergesort

c. Use the tree method to solve this recurrence. How does this compare to the runtime of 2-Mergesort?

3 Asymmetric Trees

Suppose we have an algorithm which divides the input of size n into three sets of sizes $n/2$, $3n/8$, and $n/8$. It takes time linear in the size of the input for its non-recursive computation, and its base cases are when $n \leq 8$. You may assume that base cases take constant time.

a. Write down the recurrence for the runtime of the algorithm.

b. Draw the first few levels of the recursion tree, labelling each vertex with its input size.

c. Give an upper bound on the runtime for our algorithm, as tight as you can.