Notes for Recitation 14

1 The Akra-Bazzi Theorem

Theorem 1 (Akra-Bazzi, strong form)**.** *Suppose that:*

$$T(x) = \begin{cases} \text{is defined} & \text{for } 0 \le x \le x_0 \\ \displaystyle\sum_{i=1}^{k} a_i T(b_i x + h_i(x)) + g(x) & \text{for } x > x_0 \end{cases}$$

where:

- $a_1, \ldots, a_k$ *are positive constants*
- $b_1, \ldots, b_k$ *are constants between 0 and 1*
- x_0 *is "large enough" in a technical sense we leave unspecified*
- $|g'(x)| = O(x^c)$ *for some* $c \in \mathbb{N}$
- $|h_i(x)| = O(x/\log^2 x)$

Then:

$$T(x) = \Theta\left(x^p \left(1 + \int_1^x \frac{g(u)}{u^{p+1}}\, du\right)\right)$$

where p *satisfies the equation* $\sum_{i=1}^{k} a_i b_i^p = 1$*.*

The only difference between the strong and weak forms of Akra-Bazzi is the appearance of this $h_i(x)$ term in the recurrence, where $h_i(x)$ represents a small change in the size of the subproblems ($O(x/\log^2 x)$). Notice that, despite the change in the recurrence, the solution $T(x)$ remains the same in both the strong and weak forms, with no dependence on $h_i(x)$! In algorithmic terms, this means that *small* changes in the size of subproblems have no impact on the asymptotic running time.

Example: Let's compare the Θ bounds for the following divide-and-conquer recurrences.

$$T_a(n) = 3T\left(\frac{n}{3}\right) + n \qquad\qquad T_b(n) = 3T\left(\left\lfloor \frac{n}{3} \right\rfloor\right) + n$$

For $T_a(n)$ we have $a_1 = 3$, $b_1 = 1/3$, $g(n) = n$, $p = 1$.

For $T_b(n)$ we have the same parameters as for $T_a(n)$, plus $h_1(n) = \lfloor n/3 \rfloor - n/3$.

Using the strong Akra-Bazzi form, the $h_1(n)$ falls out of the equation:

$$T_a(n) = T_b(n) = \Theta(n(1 + \int_1^n \frac{u}{u^2} du)) = \Theta(n \log n).$$

The addition of the ceiling operator changes the value of $n/3$ by at most 1, which is easily $O(n/\log^2 n)$. So floor and ceiling operators have no impact on the asymptotic solution to a recurrence.

2 TriMergeSort

We noted in lecture that reducing the size of subproblems is much more important to the speed of an algorithm than reducing the number of additional steps per call. Let's see if we can improve the $\Theta(n \log n)$ bound on `MergeSort` from lecture.

Let's consider a new version of `MergeSort` called `TriMergeSort`, where the size n list is now broken into *three* sublists of size $n/3$, which are sorted recursively and then merged. Since we know that floors and ceilings do not affect the asymptotic solution to a recurrence, let's assume that n is a power of 3.

1. How many comparisons are needed to merge three lists of 1 item each?

   Solution. 3. For example, a merge of lists {4},{5}, and {2} compares 4 with 5 and 4 with 2, adding 2 to the final list. Then it compares 4 with 5 and adds 4 to the final list. Finally, it appends the remaining 5. (This could be made more efficient, but let's not worry about that here.) ■

2. In the worst case, how many comparisons are needed to merge three lists of $n/3$ items, where n is a power of 3?

   Solution. $2(n-2)+1$. The worst case occurs if the first list empties when there is exactly 1 item in each of the other two. Prior to this, each of the other $n-2$ numbers requires 2 comparisons before going into the big list. After this, we only need 1 more comparison between the 2 leftover items. ■

3. Define a divide-and-conquer recurrence for this algorithm. Let $T(n)$ be the number of comparisons to sort a list of n items.

   Solution. $T(n) = 3T(n/3) + 2n - 3$. ■

4. We could analyze the running time of this using plug-and-chug, but let's try Akra-Bazzi. First, what is p?

Solution. $p = 1$. Using $\sum_{i=1}^{k} a_i b_i^p = 1$ with $a_1 = 3$ and $b_1 = 1/3$, we get the constraint that $3(1/3)^p = 1$. ■

5. Does the condition $|g'(x)| = O(x^c)$ hold for some $c \in N$?

   Solution. Yes. $g(n) = 2n - 3$, so $|g'(n)| = 2 = O(n^c)$ for $c = 0$. ■

6. Determine the theta bound on T(n) by integration.

   Solution.

$$\begin{aligned}
T(n) =& \Theta\left(n^p\left(1 + \int_1^n \frac{g(u)}{u^{p+1}}\,du\right)\right)\\
=& \Theta\left(n\left(1 + \int_1^n \frac{2u-3}{u^2}\,du\right)\right)\\
=& \Theta\left(n\left(1 + \int_1^n \frac{2}{u} - \int_1^n \frac{3}{u^2}\,du\right)\right)\\
=& \Theta\left(n\left(1 + 2\log u\Big|_1^n + \frac{3}{u}\Big|_1^n\right)\right)\\
=& \Theta\left(n\left(1 + 2\log n + \frac{3}{n} - 3\right)\right)\\
=& \Theta\left(2n\log n - 2n + 3\right)\\
=& \Theta\left(n\log n\right)
\end{aligned}$$

■

7. Turns out that any equal partition of the list into a constant number of sublists $c > 1$ will yield the same theta bound. Can you see why?

   Solution. Given a constant size partition, the recurrence will always be in the form:

$$T(n) = cT(n/c) + \left[(c-1)(n-(c-1)) + \sum_{i=1}^{c-2} i\right]$$

   The first term creates the constraint that $c(1/c)^p = 1$, which always gives $p = 1$. The second term will always be $\Theta(n)$, which dominates the final bound after integration. Therefore, no matter what $c > 1$ we choose, $T(n) = \Theta(n \log n)$. ■

Guessing a Particular Solution

A general linear recurrence has the form:

$$f(n) = b_1 f(n-1) + b_2 f(n-2) + \ldots + b_d f(n-d) + g(n)$$

One step in solving this recurrence is finding a *particular solution*. This is a function $f(n)$ that satisfies the recurrence equation, but may not be consistent with the boundary conditions. Here's a recipe to help you guess a particular solution:

- If $g(n)$ is a constant, guess that $f(n)$ is some constant c. Plug this into the recurrence equation and see if any constant actually works. If not, try $f(n) = bn + c$, then $f(n) = an^2 + bn + c$, etc.
- More generally, if $g(n)$ is a polynomial, try a polynomial of the same degree. If that fails, try a polynomial of degree one higher, then two higher, etc. For example, if $g(n) = n$, then try $f(n) = bn + c$ and then $f(n) = an^2 + bn + c$.
- If $g(n)$ is an exponential, such as 3^n, then first guess that $f(n) = c3^n$. Failing that, try $f(n) = bn3^n + c3^n$ and then $an^2 3^n + bn3^n + c3^n$, etc.

In practice, your first or second guess will almost always work.

Linear Recurrences

Find closed-form solutions to the following linear recurrences.

1. $T_0 = 0$
   $T_1 = 1$
   $T_n = T_{n-1} + T_{n-2} + 1$

   Solution. Following the guide to solving linear recurrences:

   (a) First, we find the general solution to the homogeneous recurrence. The characteristic equation is $r^2 - r - 1 = 0$. The roots of this equation are:

   $$\begin{aligned} r_1 &= \frac{1+\sqrt{5}}{2} \\ r_2 &= \frac{1-\sqrt{5}}{2} \end{aligned}$$

   (b) Using the roots, we write down the homogeneous solution in the form

   $$T_n = A\left(\frac{1+\sqrt{5}}{2}\right)^n + B\left(\frac{1-\sqrt{5}}{2}\right)^n.$$

   (c) Next, we need a particular solution to the inhomogenous recurrence. Since the inhomogenous term is constant, we guess a constant solution, c. So replacing the $T(n)$ terms in $T_n = T_{n-1} + T_{n-2} + 1$ by c, we require

   $$c = c + c + 1,$$

   namely, $c = -1$. That is, $T_n = -1$ is a particular solution to the equation.

   (d) Putting it together, the complete solution to the recurrence is the homogenous solution plus the particular solution:

   $$T_n = A\left(\frac{1+\sqrt{5}}{2}\right)^n + B\left(\frac{1-\sqrt{5}}{2}\right)^n - 1$$

   (e) All that remains is to find the constants A and B. Substituting the initial conditions gives a system of linear equations.

   $$\begin{aligned} 0 &= A + B - 1 \\ 1 &= A\left(\frac{1+\sqrt{5}}{2}\right) + B\left(\frac{1-\sqrt{5}}{2}\right) - 1 \end{aligned}$$

The solution to this linear system is:

$$\begin{aligned} A &= \frac{5+3\sqrt{5}}{10} \\ B &= \frac{5-3\sqrt{5}}{10} \end{aligned}$$

Therefore, the complete solution to the recurrence is

$$T_n = \left(\frac{5+3\sqrt{5}}{10}\right)\cdot\left(\frac{1+\sqrt{5}}{2}\right)^n + \left(\frac{5-3\sqrt{5}}{10}\right)\cdot\left(\frac{1-\sqrt{5}}{2}\right)^n - 1.$$

■

2. $S_0 = 0$
   $S_1 = 1$
   $S_n = 6S_{n-1} - 9S_{n-2}$

Solution. The characteristic polynomial is $r^2 - 6r + 9 = (r-3)^2$, so the solution is of the form $A3^n + Bn3^n$ for some constants A and B. Setting $n = 0$, we have $0 = S_0 = A3^0 + B\cdot 0\cdot 3^0 = A$. Setting $n = 1$, we have $1 = S_1 = A3^1 + B\cdot 1\cdot 3^1 = 3B$, so $B = 1/3$. That is,

$$S_n = 0\cdot 3^n + \frac{1}{3}\cdot n3^n = n3^{n-1}.$$

■

6.042J / 18.062J Mathematics for Computer Science
Fall 2010