

Notes for Recitation 13

1 The Akra-Bazzi Theorem

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

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

where:

- a_1, \dots, a_k are positive constants
- b_1, \dots, 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.

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

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:

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

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 Plug & Chug

Suppose you put \$1000 in a bank account. At the end of each month, you earn 1% interest and then you immediately withdraw \$5. Let M_n be the amount of money in the account after n months.

- Express the amount in the account after n months with a recurrence and base cases.

Solution.

$$\begin{aligned} M_0 &= 1000 \\ M_n &= 1.01M_{n-1} - 5 \end{aligned} \quad (n \geq 1)$$

■

- Now we're going to find a closed form for M_n using the "plug and chug" method. Rewrite M_n in terms of smaller and smaller M_i by applying the recurrence equation over and over. Stop when you uncover a pattern. Simplify enough to keep the expressions manageable, but not so much that you destroy the pattern!

Solution.

$$\begin{aligned} M_n &= 1.01M_{n-1} - 5 \\ &= 1.01(1.01M_{n-2} - 5) - 5 \\ &= 1.01^2M_{n-2} - 5 \cdot 1.01 - 5 \\ &= 1.01^2(1.01M_{n-3} - 5) - 5 \cdot 1.01 - 5 \\ &= 1.01^3M_{n-3} - 5 \cdot 1.01^2 - 5 \cdot 1.01 - 5 \\ &= 1.01^3(1.01M_{n-4} - 5) - 5 \cdot 1.01^2 - 5 \cdot 1.01 - 5 \\ &= 1.01^4M_{n-4} - 5 \cdot 1.01^3 - 5 \cdot 1.01^2 - 5 \cdot 1.01 - 5 \end{aligned}$$

■

3. Based on the pattern you observed, what expression would you have after k rounds of plug-and-chug?

Solution.

$$M_n = 1.01^k M_{n-k} - 5 \cdot 1.01^{k-1} - \dots - 5 \cdot 1.01 - 5$$

■

4. Use your expression from the previous part, which is written in terms of k , to write M_n entirely in terms of the base cases.

Solution. Choosing $k = n$ gives:

$$M_n = 1.01^n M_0 - 5 \cdot 1.01^{n-1} - 5 \cdot 1.01^{n-2} - \dots - 5 \cdot 1.01 - 5$$

■

5. Find a closed-form for M_n by applying summation techniques to your expression and substituting in base cases.

Solution. We use the formula for the sum of a geometric series.

$$\begin{aligned} M_n &= 1.01^n M_0 - 5 \cdot \left(\frac{1 - 1.01^n}{1 - 1.01} \right) \\ &= 1.01^n(1000) + 500(1 - 1.01^n) \\ &= 500(1.01^n) + 500 \end{aligned}$$

■

3 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 $c \in \mathbb{N}$?

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.

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

■

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, $T(n) = \Theta(n \log n)$. ■