Notes for Recitation 6 (10/16/98)

1 The Akra-Bazzi Method

The Merge Sort recurrence and the variants we considered are called Divide and Conquer recurrences because they arise all the time in the analysis of Divide and Conquer algorithms.

Definition 1.1 *A* general Divide and Conquer recurrence *has the form*

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

Here x is any non-negative real number; it need not be a power of two or even an integer. In addition, a_i and x_0 are positives constants, all b_i are constants between 1 and 0, and the number of terms in the summation is a constant k.

Theorem 1.2 (Akra-Bazzi '96) *Set p so that $\sum_{i=1}^{k} a_i b_i^p = 1$. If $|g'(x)| \leq x^{O(1)}$, then*

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

1.1 The Irrelevance of Small Changes

We produced the Merge Sort recurrence in lecture only under the assumption that the number of items to be sorted was a power of two. The Akra-Bazzi solution has no such limitation. However, we must proceed carefully; if n is not a power of 2, then not only is the solution in doubt, but the recurrence itself does not describe the algorithm. If Merge Sort is called with an odd-length list, then we can not generate two subproblems of exactly equal size. To account for this, we must correct the Merge Sort recurrence as follows:

$$T(n) = T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + n - 1$$

This recurrence does not fit the form of the theorem! Though the introduction of floor and ceiling operations alters the size of subproblems by at most 1, we have seen that small changes in a recurrence can radically alter the solution. Fortunately, this particular change does not affect the Akra-Bazzi result at all. This is made precise the following theorem, which is proven in the handout. The key feature of this theorem is the new term $h_i(x)$.

Theorem 1.3 *The solution to:*

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

is given by:

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

provided that for all i, $|h_i(x)| = O(\frac{x}{\log^2 x})$.

Now we can solve the revised Merge Sort recurrence. We can rewrite the recurrence as:

$$T(n) = T(n/2 + (\lceil n/2 \rceil - n/2)) + T(n/2 + (\lfloor n/2 \rfloor - n/2)) + n - 1$$

This is just the original Merge Sort recurrence with small correction factors. We can set:

$$\begin{aligned} h_1(n) &= \lceil n/2 \rceil - n/2 \\ h_2(n) &= \lfloor n/2 \rfloor - n/2 \end{aligned}$$

Since $|h_1(n)| < 1 = O(\frac{n}{\log^2 n})$ and the same holds for $h_2(n)$, we can apply the preceding theorem. In this case, $a_1 = a_2 = 1$ and $b_1 = b_2 = 1/2$, and so $p = 1$. We have to check that g does not change too rapidly: $|g'(n)| = 1 = n^{O(1)}$.

A short computation gives $T(n) = \Theta(n(1 + \log n)) = \Theta(n \log n)$. This is the same solution as before; the floor and ceiling made no difference.

In fact, even a recurrence such as $T(n) = 2T(n/2 + \sqrt{n}) + n$ has the same solution, since $\sqrt{n} = O(\frac{n}{\log^2 n})$. The rule of thumb is that *small* changes inside $T(\ )$ do not alter the solution to Divide and Conquer recurrences.

Small changes to the additive term $g(x)$ also do not matter. (Recall that in the Merge Sort recurrence, $g(x)$ corresponds to the number of additional steps per recursive call.) In particular, suppose that $g(x)$ were multiplied by a constant factor c. Looking at the Akra-Bazzi solution, we see that the constant c passes out of the integral and is absorbed into the Θ.

$$
\begin{aligned}
T(x) &= \Theta(x^p(1+\int_1^x \frac{cg(u)}{u^{p+1}}\,du)) \\
&= \Theta(x^p(1+c\int_1^x \frac{g(u)}{u^{p+1}}\,du)) \\
&= \Theta(x^p(1+\int_1^x \frac{g(u)}{u^{p+1}}\,du))
\end{aligned}
$$

Therefore, if we do not care about constant factors, then we can write a handy rule as follows:

$$
\left.\begin{array}{l} T(n) = 2T(n/2) + \Theta(n) \\ T(1) = \Theta(1) \end{array}\right\} \Rightarrow T(n) = \Theta(n \log n)
$$

2 Change of variable

Another useful method to make a difficult looking recurrence easier is by a change of variable. This is similar to the change of variable technique in calculus. For example, consider the recurrence

$$T(n) = 2T(\lfloor\sqrt{n}\rfloor) + \log n$$

with base condition $T(1) = 1$. Here $T(n)$ is defined for all n. Let us first note that $T(n)$ is a nondecreasing function of n.

Claim 2.1 *For all $n > 1$, $T(n) \geq T(n-1)$.*

Proof. By strong induction on n.

I.H.: $P(n) : T(n) \geq T(n-1)$.

Base: $n = 2$: $T(2) = 2T(\lfloor\sqrt{2}\rfloor) + 1 = 2T(1) + 1 = 3 > 1 = T(1)$.

Ind. Step: Assume $P(k)$ is true for all $k \leq n$. We will prove that $P(n+1)$ is true.

$$
\begin{aligned}
T(n+1) &= 2T(\lfloor\sqrt{n+1}\rfloor) + \log(n+1) \\
&\geq 2T(\lfloor\sqrt{n}\rfloor) + \log n
\end{aligned}
$$

The latter is true because either $\lfloor\sqrt{n+1}\rfloor = \lfloor\sqrt{n}\rfloor$ or we can apply $P(\lfloor\sqrt{n+1}\rfloor)$ which says that $T(\lfloor\sqrt{n+1}\rfloor) \geq T(\lfloor\sqrt{n}\rfloor)$.

But then the right hand side is $T(n)$. So $T(n+1) \geq T(n)$. ■

We will now solve the recurrence assuming n is a power of 2. First, make the substitution $n = 2^m$. The recurrence becomes

$$T(2^m) = 2T(\lfloor 2^{m/2} \rfloor) + m$$

Now define $S(m) = T(2^m)$ for all integers $m \geq 0$. Note that since T is nondecreasing, so is S. Therefore, $S(m) = T(2^m) = 2T(\lfloor 2^{m/2} \rfloor) + m \leq 2T(2^{\lceil m/2 \rceil}) + m = 2S(\lceil m/2 \rceil) + m$.

That is,

$$S(m) \leq 2S(\lceil m/2 \rceil) + m.$$

Now to bound S, we define $S'(0) = S(0), S'(1) = S(1)$, and $S'(m) = 2S'(\lceil m/2 \rceil) + m$ for $m > 1$. Now we can apply Akra Bazzi to get $S'(m) = \Theta(m \log m)$. But $S(m) \leq S'(m)$ for all m, so the $\Theta()$ bound on S' becomes an $O()$ bound on S:

$$S(m) = O(m \log m).$$

Therefore, there is a $d > 0$ such that $T(2^m) \leq d(m \log m)$ for all large m. Equivalently, $T(n) \leq d(\log n \ \log \log n)$ whenever n is a large power of two. But $\log 2n = \Theta(\log n)$ and T is nondecreasing, so we can conclude that the bound of a constant times $\log n \log \log n$ holds for even for n *between* powers of two. In other words,

$$T(n) = O(\log n \ \log \log n).$$

What about a lower bound? This can be derived in a similar fashion.

$$S(m) = T(2^m) = 2T(\lfloor 2^{m/2} \rfloor) + m \geq 2T(2^{\lfloor m/2 \rfloor}) + m = 2S(\lfloor m/2 \rfloor) + m.$$

That is, $S(m) \geq 2S(\lfloor m/2 \rfloor) + m$.

This time define $S'(0) = S(0)$ and $S'(m) = 2S'(\lfloor m/2 \rfloor) + m$ for $m > 0$, so $S(m) \geq S'(m)$ for all m. Now we can again apply Akra-Bazzi to solve the recurrence for S' to get $S'(m) = \theta(m \log m)$, so $m \log m = O(S(m))$. By reasoning similar to that above for the $O()$ bound, we conclude that

$$\log n \ \log \log n = O(T(n)).$$

Combining the upper and lower bounds, we have shown

$$T(n) = \Theta(\log n \log \log n).$$

The hard part in changing variables to solve recurrences is guessing what substitution to make, but this is a skill that comes out of practice, just as in calculus.

3 Linear Recurrences

3.1 Tricky Case: Repeated Roots

If the characteristic equation of a recurrence has a repeated root, then the solution is slightly more complicated. The main tool is the following theorem, which we will not prove.

Theorem 3.1 *If r is a root of the characteristic equation of a recurrence with multiplicity m, then all of $r^n, nr^n, n^2r^n, \ldots n^{m-1}r^n$ are solutions to the recurrence.*

For example, if the characteristic equation of a recurrence has root r_1 with multiplicity 2, root r_2 with multiplicity 1, and root r_3 with multiplicity 3, then the solution to the recurrence has the form:

$$f(n) = c_1r_1^n + c_2nr_1^n + c_3r_2^n + c_4r_3^n + c_5nr_3^n + c_6n^2r_3^n$$

As before, the constants depend on the boundary conditions. Each boundary condition generates a linear equation, and the constants are computed by solving the resulting system of linear equations. In this respect, solving linear recurrences is very similar to solving linear differential equations.

3.2 An Example of the Tricky Case

To understand recurrences of this tricky type, let's work an example.

Suppose that there is a type of plant that lives forever, but only reproduces during the first year of life. How fast will the plant population grow? (Notice that this is just the reverse of the graduate student job problem where faculty "reproduce" in every year except the first.)

Let $f(n)$ be the number of plants in year n. As boundary conditions, define $f(0) = 0$ and $f(1) = 1$. The plant population in year n is equal to the population from the year before plus the number of new plants. The population from the year before is $f(n-1)$. The number of new plants this year is equal to the number of new plants last year or $f(n-1) - f(n-2)$. We can now form a recurrence equation for the plant population:

$$\begin{aligned} f(n) &= f(n-1) + (f(n-1) - f(n-2)) \\ &= 2f(n-1) - f(n-2) \\ f(n) - 2f(n-1) + f(n-2) &= 0 \end{aligned}$$

The characteristic equation is $\alpha^2 - 2\alpha + 1 = 0$, which has the single root $\alpha = 1$ with multiplicity 2. Therefore, the solution to the recurrence has the form:

$$\begin{aligned} f(n) &= c_1(1)^n + c_2 n(1)^n \\ &= c_1 + c_2 n \end{aligned}$$

The boundary conditions, $f(0) = 0$ and $f(1) = 1$, generate the following linear equations:

$$\begin{aligned} 0 &= c_1 + c_2(0) \\ 1 &= c_1 + c_2(1) \end{aligned}$$

The solution to the linear system is $c_1 = 0$ and $c_2 = 1$. Therefore, the solution to the recurrence is:

$$\begin{aligned} f(n) &= c_1 + c_2 n \\ &= 0 + (1)n \\ &= n \end{aligned}$$

The answer turns out to be very simple! In year n, there are exactly n plants. Of course, we probably could have solved this problem more easily with Guess and Verify, but for many other examples, guessing is not so easy.

3.3 An Example of an Inhomogeneous Recurrence

Let $f(n) = 4f(n-1) + 3^n$ and, as a boundary condition, let $f(1) = 1$. This is an inhomogeneous linear recurrence.

Step 1: Solve the Homogeneous Recurrence

The homogeneous recurrence is $f(n) - 4f(n-1) = 0$. The characteristic equation is $\alpha - 4 = 0$. The only root is $\alpha = 4$. Therefore, the homogeneous solution is $c_1 4^n$.

Step 2: Find a Particular Solution

We must find a single solution to the recurrence $f(n) = 4f(n-1)+3^n$. This is usually the hard part of solving an inhomogeneous linear recurrence. Let's guess that there is a solution of the form $c3^n$. Substituting this into the recurrence gives:

$$\begin{aligned} f(n) &= 4f(n-1) + 3^n \\ c3^n &= 4c3^{n-1} + 3^n \\ 3c &= 4c + 3 \\ c &= -3 \end{aligned}$$

The equation was divided by 3^{n-1} in the first step. The second and third steps were only simplification. We now have a particular solution, $f(n) = -3 \cdot 3^n = -3^{n+1}$.

Step 3: Add Solutions and Find Constants

We now add the homogeneous solution and the particular solution to obtain the general solution:

$$f(n) = c_1 4^n - 3^{n+1}$$

The value of the constant is obtained from the boundary condition, $f(1) = 1$:

$$\begin{aligned} 1 &= c_1 4^1 - 3^{1+1} \\ &= 4c_1 - 9 \\ c_1 &= 5/2 \end{aligned}$$

The solution to the recurrence is $f(n) = \frac{5}{2}4^n - 3^{n+1}$!

Since we could easily have made a mistake, let's check to make sure that our solution at least works for $n = 2$. From the recurrence, $f(2) = 4f(1) + 3^2 = 13$. From our closed form, $f(2) = \frac{5}{2}4^2 - 3^3 = 40 - 27 = 13$. It looks right!

Team Problems

Problem 1. Life and Death. In ancient Mars, cannibalism was an essential way of life. The Marons and the Batons were two species that lived for several thousand years in close contact. Every year, first all the Marons sustained themselves by first consuming Batons: two Marons ate one Baton between them. Then they reproduced: one-year-old Marons doubled in number while Marons that were two or older tripled in number. Then the surviving Batons consumed one Maron each and then all the Batons reproduced to quadruple in number. This cycle repeated every year. After 1 year there were two 1-year old Marons and two 1-year old Batons. After 2 years there were two 1-year olds and two 2-year olds in each of the two species. How many of each will there be of each species at the end of n years?

Solution: Let $g(n)$ be the number of Marons after n years, and $f(n)$ the number of Batons.

After $n-1$ years, we have $g(n-1)$ Marons and $f(n-1)$ Batons. Then in the n^{th} year, from the rules of sustenance, the population of Batons drops to $f(n-1)-(1/2)g(n-1)$. The number of 1-year old Marons is $g(n-1)-g(n-2)$. Their population goes up to $2(g(n-1)-g(n-2))$. Meanwhile the number of older Marons goes up from $g(n-2)$ to $3g(n-2)$. Then it is the Batons turn. The survivors consume a Maron each. So the population of Marons after n years is

$$g(n) = 2(g(n-1)-g(n-2)) + 3g(n-2) - (f(n-1) - \frac{1}{2}g(n-1))$$

That is,

$$g(n) = \frac{5}{2}g(n-1) + g(n-2) - f(n-1)$$

The Batons then quadruple. So the number of Batons after n years is:

$$f(n) = 4(f(n-1) - \frac{1}{2}g(n-1))$$

We know that $f(1) = g(1) = 2$ and $f(2) = g(2) = 4$. Substituting these values in the recurrences, we get $f(3) = g(3) = 8$ and $f(4) = g(4) = 16$.

We guess that $g(n) = f(n) = 2^n$. We can verify this by strong *double* induction. Note that there is only 1 variable, n, the number of years, but we will argue inductively about both $f(n)$ and $g(n)$.

I.H. : $P(n) : f(n) = g(n) = 2^n$.

Base: $n = 1$: $f(1) = g(1) = 2$ and

$n = 2$: $f(2) = g(2) = 4$.

Ind. Step.: Assume $P(k)$ is true for $1 \leq n$. To prove $P(n+1)$ consider $f(n+1), g(n+1)$.

$$f(n+1) = 4(f(n) - \frac{1}{2}g(n)) = 4(2^n - 2^{n-1})$$
$$= 4(2^{n-1}) = 2^{n+1}$$

And,

$$g(n+1) = \frac{5}{2}g(n) + g(n-1) - f(n) = \frac{5}{2}(2^n) + 2^{n-1} - 2^n = 2^{n+1}$$

Problem 2. **S**tepping up. Alice the "elegant" climbs up a flight of stairs one or two stairs in one stride. Bob the "elongated beast" on the other hand clambers up 2, 4, 6 or any even number of stairs in one stride. Who can climb a flight of n stairs in more ways? (Assume n is even. A *way* of climbing a flight of n stairs is the sequence of strides taken to get to the top. E.g. to climb up 4 stairs, Alice can go 1,1,1,1 or 1,1,2, or 1,2,1, or 2,1,1 or 2,2 i.e., in 5 different ways in all.)

Solution: Let $A(n)$ be the number of ways Alice can climb up a flight of n stairs. Then, using the fact that her last stride is a 1-step or a 2-step,

$$A(n) = A(n-1) + A(n-2)$$

where $A(1) = 1, A(2) = 2, \ldots$. This is the familiar Fibonacci recurrence, So $A(n) = F_{n+1}$, the $(n+1)^{th}$ Fibonacci number.

Similarly Let $B(n)$ be the number of ways Bob can climb n stairs. Assume n is even.

$$B(n) = B(n-2) + B(n-4) + \ldots + B(2)$$

From this we can say that for $n \geq 4$,

$$B(n-2) = B(n-4) + \ldots + B(2)$$

Subtracting the latter equation from the former,

$$B(n) - B(n-2) = B(n-2)$$

$$B(n) = 2B(n-2)$$

We can solve this by expanding the recurrence relation repeatedly:

$$B(n) = 2B(n-2) = 4B(n-4) = 2^3 B(n-6) = 2^k B(n-2k) = 2^{n/2-1} B(2)$$

$$B(n) = 2^{n/2-1}$$

But then $B(n) \leq F_{n+1} = A(n)$.