CMSC 351 - Introduction to Algorithms
Spring 2012
Lecture 5

Instructor: MohammadTaghi Hajiaghayi
Scribe: Rajesh Chitnis

1 Introduction

In this lecture we look at Recurrence Relations.

2 Recurrence Relations

A **recurrence relation** is a way to define a function by an expression involving the same function, i.e., defining the function inductively.

The most famous example of a recurrence relation are the Fibonacci numbers given by $F(n) = F(n-1) + F(n-2), F(0) = 1, F(1) = 1$. We should have enough information to obtain $F(n)$ inductively, i.e., the base case and the rest of the inductive case.

Note that in the above example, to obtain $F(n)$ we need to compute all the values $F(1), F(2), \ldots, F(n-1)$ which takes $O(n)$ time. It would be much easier if we have an explicit (or closed-form) expression for $F(n)$. This is called as solving the recurrence relation, e.g., for the n^{th} Fibonacci we can get the following explicit formula:

$$F(n) = \frac{\varphi^n - \psi^n}{\sqrt{5}} \tag{1}$$

where $\varphi = \frac{1+\sqrt{5}}{2} \approx 1.6180$ (golden ratio) and $\psi = \frac{1-\sqrt{5}}{2} = -0.618033$.

3 Solving Recurrence Relations

Recurrence relations are used quite often in the analysis of algorithms and hence we learn how to solve them.

3.1 Intelligent Guesses

Guessing and playing with the recurrence relation works well for wide class of recurrence relations. Of course this needs some experience and practice of working with recurrence relations. Substituting and expanding is another technique which is used quite often. Let us now see some standard forms.

1. $a_n = ra_{n-1}$ and $a_0 = k$
   Then we have $a_1 = rk, a_2 = r^2k$ and in general $a_n = r^nk$

2. $a_n = Aa_{n-1} + Ba_{n-2}$; $a_0 = k$ and $a_1 = h$
   Let us guess $a_n = r^n$ and substitute it in the formula. This gives $r^n = Ar^{n-1} + Br^{n-2} \Rightarrow r^2 = Ar + B$. We solve this to get two roots r_1 and r_2. Now it is easy to see that Cr_1^n and Dr_2^n and more generally $Cr_1^n + Dr_2^n$ (linear combination) if $r_1 \neq r_2$, and $Cr_1^n + nDr_2^n$ if $r_1 = r_2$ are also the solutions and indeed are the most general solutions. The constants C and D are chosen based on the two given initial values a_0 and a_1, i.e., $C+D = k$ and $Cr_1 + Dr_2 = h$ (solve the exact equations to obtain C and D). For the Fibonacci numbers, if we solve $C + D = 1$ and $Cr_1 + Dr_2 = 1$ then we obtain the explicit formula given in Equation 1. The equation $r^2 = Ar+B$ is called as the **characteristic** equation and this same technique can also be used for $a_n = A_1a_{n-1} + A_2a_{n-2} + \ldots + A_ka_{n-k}$. Again we note that proof would be by induction. See Wikipedia for more formulae.

3. $a_n = a_{n-1} + n$ and $a_0 = k$
   Then

$$\begin{aligned} a_n &= a_{n-1} + n \\ &= a_{n-2} + (n-1) + n \\ &= \ldots \\ &= a_0 + 1 + 2 + \ldots + n \\ &= k + \frac{n(n+1)}{2} \end{aligned}$$

   This is an example of proof by **expansion**.

4. $a_n = 2a_{\lfloor \frac{n}{2} \rfloor} + n$ and $a_2 = 2$
   Sometimes in the running times we have $a_n \leq 2a_{\lfloor \frac{n}{2} \rfloor} + n$ but we make it equality since we care only about the upper bound (O notation). First we drop $\lfloor\rfloor$ and thus $a_n = 2a_{\frac{n}{2}+n}$ or alternatively assume that n is even. Let $n = 2^k$ and thus $k = \log n$, thus $a_{2^k} = 2a_{2^{k-1}} + 2^k$. Let $b_k = a_{2^k}$ and thus $b_1 = a_2$. Hence $b_k = 2b_{k-1} + 2^k$. This is called as the **substitution**

method. Then

$$\begin{aligned}
a_n &= a_{2^k} \\
&= b_k \\
&= 2b_{k-1} + 2^k \\
&= 2(2b_{k-2} + 2^{k-1}) + 2^k \\
&= 2^2 b_{k-2} + (2 \times 2^k) \\
&= 2^3 b_{k-3} + (3 \times 2^k) \\
&= \dots \\
&= 2^{k-1} b_1 + (k-1)2^k \\
&= 2^{k-1} a_2 + (\log n - 1)2^k \\
&= n + n(\log n - 1) \\
&= n \log n
\end{aligned}$$

Thus $a_n = O(n \log n)$. Now we can come back and prove inductively that $a_n \leq cn \log n$ for some constant c. The base case is $2 = a_2 \leq c(2 \times 1) = 2c$ which holds for all $c \geq 1$. Induction Hypothesis is $a_{\lfloor \frac{n}{2} \rfloor} \leq c\lfloor \frac{n}{2} \rfloor \log \lfloor \frac{n}{2} \rfloor$, then $a_n = 2a_{\lfloor \frac{n}{2} \rfloor} + n \leq 2c\lfloor \frac{n}{2} \rfloor \log \lfloor \frac{n}{2} \rfloor + n \leq 2c\frac{n}{2} \log \frac{n}{2} + n \leq cn(\log n - 1) + n = cn \log n - cn + n \leq cn \log n$ for all $c \geq 1$. So the induction works for any $c \geq 1$.

Indeed the formula above is the number of comparisons in a procedure for sorting n numbers, called the Mergesort: We divide the sequence into two parts of (almost) equal size ($\lfloor \frac{n}{2} \rfloor, \lceil \frac{n}{2} \rceil$). We sort them and then merge them with n more comparisons. Thus we have the recurrence given above.

References

[1] Udi Manber, *Introduction to Algorithms - A Creative Approach*