Recurrence Relations

Help Center

A *recurrence relation* is a set of equations that defines a function of one or more numbers recursively. Typically, initial values are given for the function (*base cases*) with larger values being defined as a recursive function of smaller values. A simple example might be something of the form:

$$f(1) = 1$$
$$f(n) = f(\tfrac{n}{2}) + 1$$

Typically, recurrences are used as a model for various types of mathematical processes. Once the process is expressed as a recurrence, determining a closed form expression for the function in terms of its parameter(s) is often as simple as looking up a solution. For example, the recurrence above has a solution of the form $f(n) \approx \log_2(n) + 1$.

Modeling the behavior of recursive programs using recurrences

In Computer Science, one important use of recurrences is that they provide a simplified model of the behavior of recursive functions. For example, consider problem of estimating the number of comparisons of the value `item` to elements in `ordered_list` during this binary search. Let $f(n)$ be the number of comparisons required for a list of size n. Now, note that one call to a binary search on a list of size n requires one comparison plus the number of comparisons required to perform a binary search on a list of size $\frac{n}{2}$, which is $f(\frac{n}{2})$. Thus, the unknown function $f(n)$ satisfies the example recurrence above.

As another example, consider the problem of determining the number of binary numbers of length n as computed in this recursive program. Again, let $f(n)$ denote the number of binary numbers of length n. The recursive program computes the list of binary numbers of length $n-1$ which would have length $f(n-1)$. Then, the program makes two copies of this list (appending `"0"` and `"1"` to each element) to create the desired list. This leads to recurrence on $f(n)$ of the form:

$$f(0) = 1$$
$$f(n) = 2\,f(n-1)$$

This recurrence has the solution $f(n) = 2^n$ which agrees with the observation that there are 2^n n-bit binary numbers.

Solutions to common recurrences

When building recursive programs, knowing the closed-form solutions to common recurrences is helpful since this knowledge can guide you towards building efficient programs. Below, we list various common recurrences and their approximate solutions. In all cases, we assume that $f(1) = 1$.

- $f(n) = f(n-1) + 1 \quad \rightarrow \quad f(n) = n$
- $f(n) = f(n-1) + n \quad \rightarrow \quad f(n) = \frac{1}{2}\,n(n+1)$
- $f(n) = 2f(n-1) \quad \rightarrow \quad f(n) = 2^{n-1}$
- $f(n) = n\,f(n-1) \quad \rightarrow \quad f(n) = n!$
- $f(n) = f(\frac{n}{2}) + 1 \quad \rightarrow \quad f(n) \approx \log_2(n) + 1$
- $f(n) = f(\frac{n}{2}) + n \quad \rightarrow \quad f(n) \approx 2n - 1$

- $f(n) = 2f(\frac{n}{2}) \quad \rightarrow \quad f(n) \approx n$
- $f(n) = 2f(\frac{n}{2}) + 1 \quad \rightarrow \quad f(n) \approx 2n - 1$
- $f(n) = 2f(\frac{n}{2}) + n \quad \rightarrow \quad f(n) \approx n(\log_2(n) + 1)$

Feel free to use this table to look up the solution to recurrences that you encounter during class. If you need to find a closed-form solution for a recurrence not in this table, you might wish to read up on the Master Theorem, which provides a solution to recurrences of the form $f(n) = a\,f(\frac{n}{b}) + g(n)$ where $g(n)$ is known. You are also welcome to plot exact and approximate solutions to the recurrences above using this code.

Created Fri 16 May 2014 11:14 AM PDT

Last Modified Wed 16 Jul 2014 11:51 AM PDT