Visualizing recurrences

The Sorcerer's Apprentice

Although the theoretical foundation of computing has been developed in the last century, the ideas behind these concepts have existed for centuries. For example, the Sorcerer's Apprentice is tale in which an old sorcerer asks his apprentice to fetch a pail of water. The apprentice, thinking himself clever, enchants a broom to do the work for him. Unfortunately for the apprentice, he has no way to stop the broom and the floor of the sorcerer's lab is soon flooded with water. Panicking, the apprentice split the broom with an ax and to his horror, the broom splits into two brooms that fetch even more water. Repeatly splitting the brooms yields an army of brooms fetching water.

This tale illustrates the power (and danger) of recursion. Given a task, we can spawn multiple subtasks to help accomplish the original task. In case of the Sorcerer's Apprentice, the number of brooms grew exponentially as the apprentice split them. For recursive algorithms, the behavior of the algorithm depends on the structure of the recursion. In this activity, we explore the use of a simulator designed to visualize the behavior of recurrences.

The structure of recurrences

For this activity, we are interested in the behavior of recurrences $r(n)$ for the form

$$r(n) = a(n)r(b(n)) + c(n)$$

where $a(n)$, $b(n)$, and $c(n)$ are simple functions of n. These three functions determine the following aspects of the behavior of the recurrence.

- $a(n)$ determines the number of recursive calls required during the evaluation of $r(n)$.
- $b(n)$ determines the size of each sub-problem evaluated during the recursive calls.
- $c(n)$ determines the amount of work required during evaluation of $r(n)$ beyond that involved in the recursive calls.

To keep the behavior of the recurrences more uniform, we always assume that $r(1) = 1$.

A simulator for visualizing recurrences

During the evaluation of a recursive function, the computing environment maintains a stack of calls to the recursive function that need to be evaluated. During this evaluation process, more calls may be added to the stack as the result of evaluating the recursive function. At some point, the calls reach the base case for the recursion and no more recursive calls are spawned.

This interactive simulator allows the user to visualize this process as the value of $r(n)$ is computed recursively. The control panel on the left-hand side displays the current recurrence and allows the user to interactively modify the structure of the recurrence using the three buttons below the recurrence. These buttons toggle through various possibilities for $a(n)$, $b(n)$, and $c(n)$, respectively, as indicated below.

- For the number of recursive calls, the simulator models $a(n) = 1$, $a(n) = 2$, and $a(n) = n$.
- For the size of the input to the recursive calls, the simulator models $b(n) = n - 1$ and $b(n) = \frac{n}{2}$.
- For the amount of work beyond that involved in the recursive calls, the simulator models $c(n) = 0$.

$c(n) = 1$ and $c(n) = n$.

There are $3 \times 2 \times 3 = 18$ possible recurrences including the nine recurrences that we consider in the math notes on recurrences.

Experimenting with recurrences

The initial instance of the recurrence $r(n)$ is visualized as a circle whose area is proportional to the value n. Clicking on this circle causes the simulator to apply the currently selected recurrence and replace the circle by $a(n)$ circles of area $b(n)$. These circles model the recursive sub-problems generated by applying the right-hand side the recurrence. Finally, the "Current work" associated with recurrence (displayed in the upper right) is incremented by the value $c(n)$.

As the user clicks on the circles, the areas of the resulting circles decrease due to the possible choices for $b(n)$. While the number of circles may increase quite substantially, eventually each circle disappears when the base case $r(1)$ is reached. When all circles are gone, "Current work" contains the value of $r(n)$.

We suggest that you experiment with the simulator for various types of recurrence to gain an intuitive feel for their rates of growth. The solutions produced by the simulator should agree with the values produced by our recurrence plotting code. The most crucial parameter to experiment with is $a(n)$. If $a(n) = 1$, the values of the resulting recurrence almost always grow quite slowly as a function of n. On the other hand, if $a(n) = n$, the values of the recurrences grow at rate that is difficult to appreciate. For an example of a recurrence that grows quite fast, we recommend experimenting with the recurrence

$$r(n) = n\, r(n-1) + n.$$

Created Wed4 Jun 2014 11:26 PM BST

Last Modified Fri 18 Jul 2014 10:19 PM BST