MAT1830

Lecture 26: Recursion

Recursion - why should you care?

- Recursion is arguably the whole point of computation.
- Understanding recursion is vital for
  - recursive algorithms
  - strings and similar data structures
  - trees and similar data structures
- Understanding induction can help you better understand what a computer is "thinking".

Just as the structure of the natural numbers supports induction as a method of proof, it supports induction as a method of definition or of computation.

When used in this way, induction is usually called *recursion*, and one speaks of a *recursive definition* or a *recursive algorithm*.

26.1 Recursive Definitions

Many well known functions $f(n)$ are most easily defined in the "base step, induction step" format, because $f(n+1)$ depends in some simple way on $f(n)$.

The induction step in the definition is more commonly called the *recurrence relation* for f, and the base step the *initial value*.

Example. The factorial $f(n) = n!$

Initial value. $0! = 1$.
Recurrence relation. $(k+1)! = (k+1) \times k!$

Many programming languages allow this style of definition, and the value of the function is then computed by a descent to the initial value.

For example, to compute $4!$, the machine successively computes

$$\begin{aligned} 4! &= 4 \times 3! \\ &= 4 \times (3 \times 2!) \\ &= 4 \times (3 \times (2 \times (1!))) \\ &= 4 \times (3 \times (2 \times (1 \times 0!))) \end{aligned}$$

which can finally be evaluated since $0! = 1$.

Remark. The numbers $4, 3, 2, 1$ have to be stored on a "stack" before the program reaches the initial value $0! = 1$ which finally enables it to evaluate $4!$.

Thus a recursive program, though short, may run slowly and even cause "stack overflow."

Flux Exercise (LQMTZZ)

The number of digits in $100!$ is

A. 3

B. 100

C. 121

D. 158

E. 200

Example. The Fibonacci sequence $\mathbf{0, 1, 1, 2, 3, 5, 8, \ldots}$

The n^{th} number $F(n)$ in this sequence is defined by

Initial values. $F(0) = 0, F(1) = 1$.
Recurrence relation. $F(k+1) = F(k)+F(k-1)$.

Remark. Using a recursive program to compute Fibonacci numbers can easily lead to stack overflow, because each value depends on two previous values (each of which depends on another two, and so on).

A more efficient way to use the recursive definition is to use three variables to store $F(k+1)$, $F(k)$ and $F(k-1)$. The new values of these variables, as k increases by 1, depend only on the three stored values, not on all the previous values.

Questions

26.1 A function $s(n)$ is defined recursively by

Initial value: $s(0) = 0$
Recurrence relation: $s(n+1) = s(n) + 2n + 1$ for $n \geq 0$

Write down the first few values of $s(n)$, and guess what function s is.

$s(1) = s(0) + 2 \times 0 + 1 = 1.$ $s(2) = s(1) + 2 \times 1 + 1 = 4.$
$s(3) = s(2) + 2 \times 2 + 1 = 9.$ $s(4) = s(3) + 2 \times 3 + 1 = 16.$
Guess that maybe $s(n) = n^2$?

26.2 Check that the function s you guessed in Question 26.1 satisfies
$s(0) = 0$ and $s(n+1) = s(n) + 2n + 1$

If $s(n) = n^2$ then $s(0) = 0^2 = 0$ and
$s(n) + 2n + 1 = n^2 + 2n + 1 = (n+1)^2 = s(n+1)$.

(This proves by induction that our guess was correct.)

26.2 Properties of recursively defined functions

These are naturally proved by induction, using a base step and induction step which parallel those in the definition of the function.

Example. For $n \geqslant 5$, 10 divides $n!$

Proof *Base step.*

$$5! = 5 \times 4 \times 3 \times 2 \times 1 = 10 \times 4 \times 3,$$

hence 10 divides 5!.

Induction step. We have to show

$$10 \text{ divides } k! \implies 10 \text{ divides } (k+1)!$$

Since $(k+1)! = (k+1) \times k!$ by the recurrence relation for factorial, the induction step is clear, and hence the induction is complete.

Example Prove that 10 divides $n!$ for each integer $n \geq 5$.

Solution Let $P(n)$ be the statement "10 divides $n!$".

Base step. $5! = 120$ and 10 divides 1200, so $P(5)$ is true.

Induction step. Suppose that $P(k)$ is true for some integer $k \geq 5$. This means that $k! = 10a$ for some integer a.

We want to prove that $P(k+1)$ is true. We want to show that $(k+1)! = 10b$ for some integer b.

$$\begin{aligned}
(k+1)! &= (k+1) \times k! \\
&= 10a \times (k+1) && \text{(by } P(k)\text{)} \\
&= 10a(k+1)
\end{aligned}$$

Because $a(k+1)$ is an integer, 10 divides $(k+1)!$. So $P(k+1)$ is true.

This proves that $P(n)$ is true for each integer $n \geq 5$.

Example. $F(0) + F(1) + \cdots + F(n) = F(n + 2) - 1$.

Proof *Base step.* $F(0) = 0 = F(2) - 1$, because $F(2) = 1$.

Induction step. We have to show

$$\begin{aligned} & F(0) + F(1) + \cdots + F(k) \\ & = F(k+2) - 1 \\ \Rightarrow \quad & F(0) + F(1) + \cdots + F(k+1) \\ & = F(k+3) - 1. \end{aligned}$$

Well,

$$\begin{aligned} & F(0) + F(1) + \cdots + F(k) \\ & = F(k+2) - 1 \\ \Rightarrow \quad & F(0) + F(1) + \cdots + F(k+1) \\ & = F(k+2) + F(k+1) - 1, \\ & \text{by adding } F(k+1) \text{ to both sides} \\ \Rightarrow \quad & F(0) + F(1) + \cdots + F(k+1) \\ & = F(k+3) - 1 \\ & \text{since } F(k+2) + F(k+1) = F(k+3) \\ & \text{by the Fibonacci recurrence relation} \end{aligned}$$

This completes the induction.

Definition The Fibonacci sequence $F(0), F(1), F(2), \ldots$ is defined by $F(0) = 1$, $F(1) = 1$ and $F(i) = F(i-1) + F(i-2)$ for all integers $i \geq 2$.

Example Prove that $F(0) + F(1) + \cdots + F(n) = F(n+2) - 1$ for all integers $n \geq 0$.

Solution Let $P(n)$ be the statement "$F(0) + F(1) + \cdots + F(n) = F(n+2) - 1$'.

Base step. $F(0) = 0$ and $F(2) - 1 = 1 - 1 = 0$, so $P(0)$ is true.

Induction step. Suppose that $P(k)$ is true for some integer $k \geq 0$. This means that $F(0) + F(1) + \cdots + F(k) = F(k+2) - 1$.

We want to prove that $P(k+1)$ is true. We want to show that $F(0) + F(1) + \cdots + F(k+1) = F(k+3) - 1$.

$$\begin{aligned} F(0) + F(1) + \cdots F(k+1) &= (F(0) + F(1) + \cdots + F(k)) + F(k+1) \\ &= F(k+2) - 1 + F(k+1) && \text{(by } P(k)\text{)} \\ &= F(k+3) && \text{(by definition of } F(k+3)\text{)} \end{aligned}$$

Because $a(k+1)$ is an integer, 10 divides $(k+1)!$. So $P(k+1)$ is true.

This proves that $P(n)$ is true for each integer $n \geq 5$.

Questions

26.3 If a sequence satisfies the Fibonacci recurrence relation,

$$f(n) = f(n-1) + f(n-2),$$

must it agree with the Fibonacci sequence from some point onward?

Answer No!

To match the Fibonacci sequence it has to satisfy the same recurrence relation, but that is not enough. It also must agree with enough consecutive terms to force all subsequent terms to agree. For the Fibonacci sequence "enough terms" means two.

To be concrete, the constant sequence $0, 0, 0, 0, \ldots$ does satisfy the recurrence $f(n) = f(n-1) + f(n-2)$; however, it doesn't match any two terms of the Fibonacci sequence. There are many, many other examples. e.g. 4,4,8,12,20,32,52,84,... does match a term of the Fibonacci sequence (which?) but never two in a row.

Flux Exercise (LQMTZZ)

Suppose that $g(1) = 1$, $g(2) = 4$ and

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

for $n > 2$. Then
$g(3) = 56$,
$g(4) = 9408$,
$g(5) = 221276160$,
$g(6) = 97926277968691200$.

How many digits are there in $g(10)$?

A. 1

B. 37

C. 55

D. 98

E. 244

26.3 Problems with recursive solutions

Sometimes a problem about n reduces to a problem about $n-1$ (or smaller numbers), in which case the solution may be a known recursively defined function.

Example. Find how many n-bit binary strings contain no two consecutive 0s.

We can divide this problem into two cases.

1. Strings which end in 1, e.g. 1101101.
   In this case, the string before the 1 (110110 here) can be any $(n-1)$ bit string with no consecutive 0s.

2. Strings which end in 0, e.g. 1011010.
   In this case, the string must actually end in 10, to avoid consecutive 0s, but the string before the 10 (10110 here) can be any $(n-2)$ bit string with no consecutive 0s.

Thus among strings with no consecutive 0s we find

1. Those with n bits ending in 1 correspond to those with $(n-1)$ bits.

2. Those with n bits ending in 0 correspond to those with $(n-2)$ bits.

Hence if we let $f(n)$ be the number of such strings with n bits we have

$$f(n) = f(n-1) + f(n-2).$$

This is the Fibonacci recurrence relation.

It can also be checked that

$$f(1) = 2 = F(3) \text{ and } f(2) = 3 = F(4),$$

hence it follows (by induction) that

$$\begin{aligned} f(n) = &\ \text{number of } n \text{ bit strings} \\ &\ \text{with no consecutive 0s} \\ = &\ F(n+2). \end{aligned}$$

Let $f(n)$ be the number of binary strings of length n with no two consecutive 0s. Let's think about $f(6)$. Imagine we already know $f(1), f(2), \ldots f(5)$.

How many ways can we build a binary string of length 6 with no two consecutive 0s?

Such a string must be of one of two types:

- it ends in 1, so it is `xxxxx1`
  where `xxxxx` is a binary string of length 5 with no two consecutive 0s.
- it ends in 0, so it is `yyyy10`
  where `yyyy` is a binary string of length 4 with no two consecutive 0s.

There are $f(5)$ strings of the first type as there are $f(5)$ options for `xxxxx`.

There are $f(4)$ strings of the second type as there are $f(4)$ options for `yyyy`.

So $f(6) = f(5) + f(4)$ because every string we're interested in is of one of the two types.

There's nothing magical about 6: in general $f(n) = f(n-1) + f(n-2)$.

26.3 Problems with recursive solutions

Sometimes a problem about n reduces to a problem about $n-1$ (or smaller numbers), in which case the solution may be a known recursively defined function.

Example. Find how many n-bit binary strings contain no two consecutive 0s.

We can divide this problem into two cases.

1. Strings which end in 1, e.g. 1101101.
   In this case, the string before the 1 (110110 here) can be any $(n-1)$ bit string with no consecutive 0s.

2. Strings which end in 0, e.g. 1011010.
   In this case, the string must actually end in 10, to avoid consecutive 0s, but the string before the 10 (10110 here) can be any $(n-2)$ bit string with no consecutive 0s.

Thus among strings with no consecutive 0s we find

1. Those with n bits ending in 1 correspond to those with $(n-1)$ bits.

2. Those with n bits ending in 0 correspond to those with $(n-2)$ bits.

Hence if we let $f(n)$ be the number of such strings with n bits we have

$$f(n) = f(n-1) + f(n-2).$$

This is the Fibonacci recurrence relation.

It can also be checked that

$$f(1) = 2 = F(3) \text{ and } f(2) = 3 = F(4),$$

hence it follows (by induction) that

$$\begin{aligned} f(n) = & \text{ number of } n \text{ bit strings} \\ & \text{ with no consecutive 0s} \\ = & \ F(n+2). \end{aligned}$$

Flux Exercise (LQMTZZ)

For all integers $n \geq 1$, let t_n be the number of ways to tile a $1 \times n$ strip using red 1×1 tiles, blue 1×2 tiles and green 1×2 tiles. Which of the following hold for integers all $n \geq 3$?

A. $t_n = t_{n-1} + t_{n-2}$

B. $t_n = 2t_{n-1} + t_{n-2}$

C. $t_n = t_{n-1} + 2t_{n-2}$

D. $t_n = 2t_{n-1} + 2t_{n-2}$

E. $t_n = 2t_{n-1}$

Example One option counted by t_5 is