Recursive definitions

Capiche?

Less is more

Recursive Definitions

1. Specify a function at its lowest/minimum level (zero? One? Empty?)
2. Give a rule for finding a value with respect to a smaller value

Sometimes called an "inductive definition"
- a base step such as f(0)
- an inductive step f(n+1) defined in terms of f(n)
- an inductive step f(n) defined in terms of f(n-1)
  - scream downhill

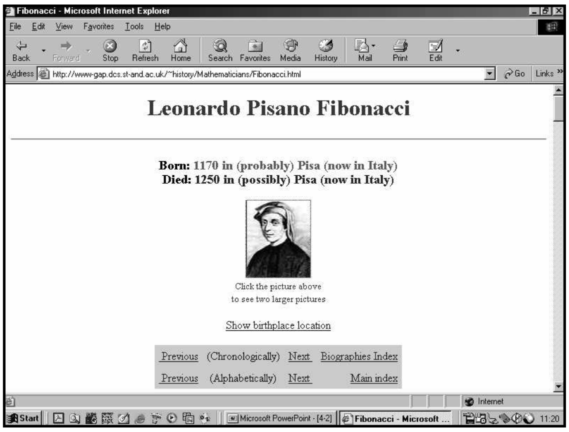
A typical example is factorial

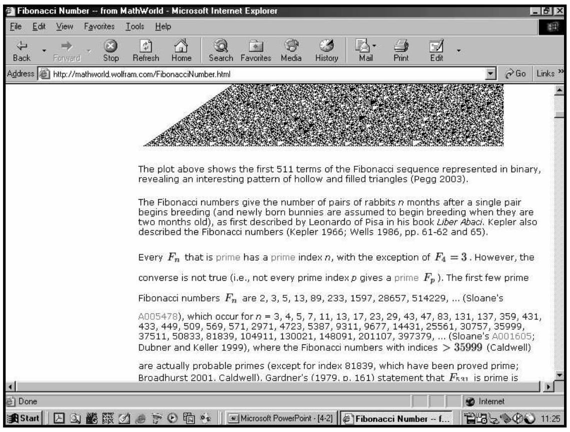
Example

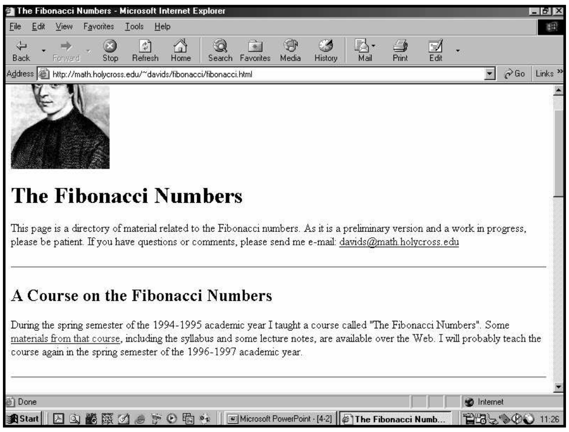
Fibonacci

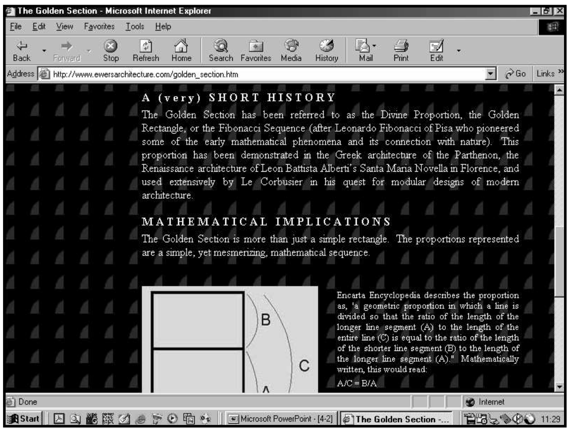
fib(n) = fib(n-1) + fib(n-2)
fib(0) = 0
fib(1) = 1

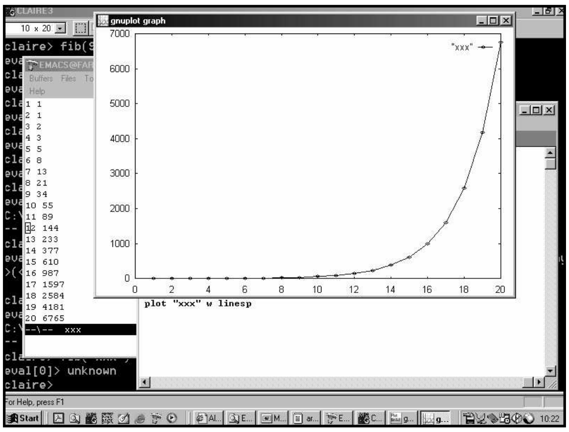
- fib(4) = fib(3) + fib(2)
- = fib(2) + fib(1) + fib(2)
- = fib(1) + fib(0) + fib(1) + fib(2)
- = 1 + 0 + fib(1) + fib(2)
- = 1 + 0 + 1 + fib(2)
- = 1 + 0 + 1 + fib(1) + fib(0)
- = 1 + 0 + 1 + 1 + 0
- = 3

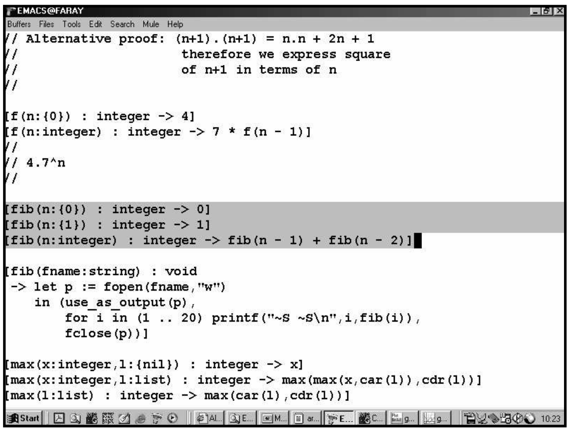

Fibonacci - Microsoft Internet Explorer
http://www-gap.dcs.st-and.ac.uk/~history/Mathematicians/Fibonacci.html
Leonardo Pisano Fibonacci
Born: 1170 in (probably) Pisa (now in Italy)
Died: 1250 in (possibly) Pisa (now in Italy)
Click the picture above
to see two larger pictures
Show birthplace location
Previous (Chronologically) Next Biographies Index
Previous (Alphabetically) Next Main index


Fibonacci Number -- from MathWorld - Microsoft Internet Explorer
http://mathworld.wolfram.com/FibonacciNumber.html
The plot above shows the first 511 terms of the Fibonacci sequence represented in binary, revealing an interesting pattern of hollow and filled triangles (Pegg 2003).
The Fibonacci numbers give the number of pairs of rabbits n months after a single pair begins breeding (and newly born bunnies are assumed to begin breeding when they are two months old), as first described by Leonardo of Pisa in his book Liber Abaci. Kepler also described the Fibonacci numbers (Kepler 1966; Wells 1986, pp. 61-62 and 65).
Every F_n that is prime has a prime index n, with the exception of $F_4 = 3$. However, the converse is not true (i.e., not every prime index p gives a prime F_p). The first few prime Fibonacci numbers F_n are 2, 3, 5, 13, 89, 233, 1597, 28657, 514229, ... (Sloane's A005478), which occur for n = 3, 4, 5, 7, 11, 13, 17, 23, 29, 43, 47, 83, 131, 137, 359, 431, 433, 449, 509, 569, 571, 2971, 4723, 5387, 9311, 9677, 14431, 25561, 30757, 35999, 37511, 50833, 81839, 104911, 130021, 148091, 201107, 397379, ... (Sloane's A001605; Dubner and Keller 1999), where the Fibonacci numbers with indices > 35999 (Caldwell) are actually probable primes (except for index 81839, which have been proved prime; Broadhurst 2001, Caldwell). Gardner's (1979, p. 161) statement that F_{531} is prime is


The Fibonacci Numbers - Microsoft Internet Explorer
http://math.holycross.edu/~davids/fibonacci/fibonacci.html
The Fibonacci Numbers
This page is a directory of material related to the Fibonacci numbers. As it is a preliminary version and a work in progress, please be patient. If you have questions or comments, please send me e-mail: davids@math.holycross.edu
A Course on the Fibonacci Numbers
During the spring semester of the 1994-1995 academic year I taught a course called "The Fibonacci Numbers". Some materials from that course, including the syllabus and some lecture notes, are available over the Web. I will probably teach the course again in the spring semester of the 1996-1997 academic year.


The Golden Section - Microsoft Internet Explorer
http://www.ewersarchitecture.com/golden_section.htm
A (very) SHORT HISTORY
The Golden Section has been referred to as the Divine Proportion, the Golden Rectangle, or the Fibonacci Sequence (after Leonardo Fibonacci of Pisa who pioneered some of the early mathematical phenomena and its connection with nature). This proportion has been demonstrated in the Greek architecture of the Parthenon, the Renaissance architecture of Leon Battista Alberti's Santa Maria Novella in Florence, and used extensively by Le Corbusier in his quest for modular designs of modern architecture.
MATHEMATICAL IMPLICATIONS
The Golden Section is more than just a simple rectangle. The proportions represented are a simple, yet mesmerizing, mathematical sequence.
B
C
A
Encarta Encyclopedia describes the proportion as, 'a geometric proportion in which a line is divided so that the ratio of the length of the longer line segment (A) to the length of the entire line (C) is equal to the ratio of the length of the shorter line segment (B) to the length of the longer line segment (A)." Mathematically written, this would read:
A/C = B/A


gnuplot graph
"xxx"
7000
6000
5000
4000
3000
2000
1000
0
0 2 4 6 8 10 12 14 16 18 20
1 1
2 1
3 2
4 3
5 5
6 8
7 13
8 21
9 34
10 55
11 89
12 144
13 233
14 377
15 610
16 987
17 1597
18 2584
19 4181
20 6765
plot "xxx" w linesp
eval[0]> unknown
claire>


EMACS@FARAY
// Alternative proof: (n+1).(n+1) = n.n + 2n + 1
//   therefore we express square
//   of n+1 in terms of n
//
[f(n:{0}) : integer -> 4]
[f(n:integer) : integer -> 7 * f(n - 1)]
//
// 4.7^n
//
[fib(n:{0}) : integer -> 0]
[fib(n:{1}) : integer -> 1]
[fib(n:integer) : integer -> fib(n - 1) + fib(n - 2)]
[fib(fname:string) : void
 -> let p := fopen(fname,"w")
    in (use_as_output(p),
        for i in (1 .. 20) printf("~S ~S\n",i,fib(i)),
        fclose(p))]
[max(x:integer,l:{nil}) : integer -> x]
[max(x:integer,l:list) : integer -> max(max(x,car(l)),cdr(l))]
[max(l:list) : integer -> max(car(l),cdr(l))]

Exercise

find f(1), f(2), f(3), f(4) and f(5), where

1. f(0) = 3, f(n+1) = -2.f(n)
   alternatively f(n) = -2.f(n - 1)
2. f(0) = 3, f(n+1) = 3.f(n) + 7
   alternatively f(n) = 3.f(n - 1) + 7

Example

Do more with less

Define arithmetic on positive integers using only

- isZero(x) : x == 0
- succ(x) : x + 1
- pred(x) : x - 1
- add(n,m) : ?
- mult(n,m) : ?
- pow(n,m) : ?
- pow2(n) : ?

Example

Define arithmetic on positive integers using only

- isZero(x) : x == 0
- succ(x) : x + 1
- pred(x) : x - 1
- add(n,m) : ?
- mult(n,m) : ?
- pow(n,m) : ?
- pow2(n) : ?

Do more with less

Example/Intuition

$$\begin{aligned} add(3,5) &= add(2,5+1) \\ &= add(1,5+1+1) \\ &= add(0,5+1+1+1) \\ &= 8 \end{aligned}$$

Example

Do more with less

Define arithmetic on positive integers using only

- isZero(x) : x == 0
- succ(x) : x + 1
- pred(x) : x - 1
- add(n,m) : if isZero(n) then m else add(pred(n),succ(m))
- mult(n,m) :
- pow(n,m) : ?
- pow2(n) : ?

$$add(n,m) = if\ \ isZero(n)\ then\ m\ else\ add(pred(n), succ(m))$$

Example

Do more with less

Define arithmetic on positive integers using only

- isZero(x) : x == 0
- succ(x) : x + 1
- pred(x) : x - 1
- add(n,m) : if isZero(n) then m else add(pred(n),succ(m))
- mult(n,m) : ?
- pow(n,m) : ?
- pow2(n) : ?

Intuition

$$5 \times 7 = 5+5+5+5+5+5+5$$

So, mult(n,m) might generate m additions of n?

Example

Do more with less

Define arithmetic on positive integers using only

- isZero(x) : x == 0
- succ(x) : x + 1
- pred(x) : x - 1
- add(n,m) : if isZero(n) then m else add(pred(n),succ(m))
- mult(n,m) : if isZero(m) then 0 else add(n,mult(n,pred(m)))
- pow(n,m) : ?
- pow2(n) : ?

$$mult(n,m) = if\ \ isZero(m)\ 0\ else\ add(n, mult(n, pred(m)))$$

$$\begin{aligned} mult(5,3) &= add(5, mult(5,2)) \\ &= add(5, add(5, mult(5,1))) \\ &= add(5, add(5, add(5, mult(5,0)))) \\ &= add(5, add(5, add(5,0))) \\ &= \ldots \end{aligned}$$

Example

Do more with less

Define arithmetic on positive integers using only

- isZero(x) : x == 0
- succ(x) : x + 1
- pred(x) : x - 1
- add(n,m) : if isZero(n) then m else add(pred(n),succ(m))
- mult(n,m) : ?
- pow(n,m) : ?
- pow2(n) : ?

$$mult(n,m) = if\ \ isZero(m)\ 0\ else\ add(n, mult(n, pred(m)))$$

$$\begin{aligned} mult(5,3) &= add(5, mult(5,2)) \\ &= add(5, add(5, mult(5,1))) \\ &= add(5, add(5, add(5, mult(5,0)))) \\ &= add(5, add(5, add(5,0))) \\ &= \ldots \end{aligned}$$

NOTE: I've assumed n and m are +ve and that m > 0

Recursion

Try and define pow(n,m) and pow2(n)

$$5! = 5\times4\times3\times2\times1$$
$$0! = 1$$

Factorial, try and define it recursively

Recursion

What have we done?
• recursively defined functions
• functions have a base case (when one argument is 1 or 0 or something)
• functions defined in terms of previous function values
• we have to make sure our recursion stops!!!

Recursive definition of a set

Recursive definition of a set

We say
• what's in the set to start with
• then how to construct new elements of the set,
  • depending on what's already in the set!

Positive integers divisible by 3

$$0 \in 3$$
$$3 \in S$$
$$x \in S \wedge y \in S \rightarrow x + y \in S$$

Notice x and y could both be the same element!
In fact, initially they must be!

Recursive definition of a set

Positive integers congruent to 2 modulo 3

$$a \equiv 2 \pmod 3$$
$$\therefore 3 \mid a-2$$
$$\therefore a-2 = 3k$$
$$\therefore a = 3k+2$$

$$2 \in S$$
$$x \in S \rightarrow 3 + x \in S$$

S = {2,5,8,11,14,17,20,...}

Exercise

Give a recursive definition of the set of positive integers that are not divisible by 5

Exercise

Give a recursive definition of the set of positive integers that are not divisible by 5

$$1 \in S, 2 \in S, 3 \in S, 4 \in S$$
$$x \in S \rightarrow 5 + x \in S$$

S = {1,2,3,4,6,7,8,9,11,12,13,14,16,...}

Recursively defined structures

Recursively defined structures

$Let\ \Sigma^*\ be\ the\ set\ of\ strings\ over\ alphabet\ \Sigma$

This can be recursively defined as follows

$$\lambda \in \Sigma^*$$
$$w \in \Sigma^* \wedge x \in \Sigma \rightarrow wx \in \Sigma^*$$

The empty string (that's lambda) is in the set

We can take some string from the set of strings and add a new character from the alphabet to the end of it

Recursively defined structures

$Let\ \Sigma^*\ be\ the\ set\ of\ strings\ over\ alphabet\ \Sigma = \{0,1\}$

$$\lambda \in \Sigma^*$$
$$w \in \Sigma^* \wedge x \in \Sigma \rightarrow wx \in \Sigma^*$$

First application generates 0 and 1
Second application generates 00, 01, 10, and 11
Third application generates 000,001,010,...,111
...

Recursively defined structures

Example, concatenation of strings

cat("abcd","hjg") = "abcdhjg"
cat("af2","") = "af2"
cat("","") = ""

$Let\ \Sigma^*\ be\ the\ set\ of\ strings\ over\ alphabet\ \Sigma$

$$w \in \Sigma^* \rightarrow w.\lambda = w$$
$$w_1 \in \Sigma^* \wedge w_2 \in \Sigma^* \wedge x \in \Sigma \rightarrow w_1.(w_2 x) = (w_1.w_2)x$$

Recursively defined structures

Example, length of a string

l("abcd") = 4
l("af2") = 3
cat("") = 0

$$l(\lambda) = 0$$
$$w \in \Sigma^* \wedge x \in \Sigma \rightarrow l(wx) = l(w) + 1$$

Note: second step is just "pattern matching", i.e. breaking up a string into a substring and its last character

$$\begin{aligned} l("paddy") &= l("padd") + 1 \\ &= l("pad") + 1 + 1 \\ &= l("pa") + 1 + 1 + 1 \\ &= l("p") + 1 + 1 + 1 + 1 \\ &= l("") + 1 + 1 + 1 + 1 + 1 \\ &= 0 + 1 + 1 + 1 + 1 + 1 \end{aligned}$$

Recursion over lists

Assume we have a list (e1,e2,e3,…,en), i.e. a list of elements

Examples:
- (ted,poppy,alice,nelson)
- (1,2,3,1,2,3,4,2,9)
- ("please","turn","the","lights","off")
- (1,dog,cat,24,crock,crock,crock)

Assume also we have the functions head(l) and tail(l)

Examples:
- l=(1,2,3,1,2,3,1); head(l) = 1; tail(l) = (2,3,1,2,3,1)
- l =("dog"); head(l) = "dog"; tail(l) = ()
- l=(); head(l) = crash!; tail(l) = crash!

Recursion over lists

$$length(()) = 0$$
$$length(l) = 1 + length(tail(l))$$

$$sum(()) = 0$$
$$sum(l) = head(l) + sum(tail(l))$$

$$exists(e, ()) = False$$
$$exists(e, l) = equals(head(l), e) \vee exists(e, tail(l))$$

Recursion over lists

Also cons(e,l) to construct a new list with element e as head and l as tail

Examples:
- cons(cat,(dog,bird,fish)) = (cat,dog,bird,fish)
- cons(3,()) = (3)
- cons(5,(4,5,4,5)) = (5,4,5,4,5)

Also snoc(l,e) to construct a new list with element e as last element of l

Examples:
- snoc((dog,bird,fish),cat) = (dog,bird,fish,cat)
- snoc((),3) = (3)
- snoc((4,5,4,5),4) = (4,5,4,5,4)

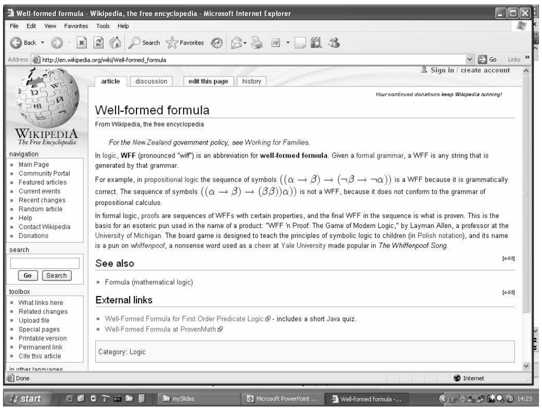
Recursion over lists

$$reverse(()) = ()$$
$$reverse(l) = snoc(tail(l), head(l))$$

NOTE: very very very inefficient

Well formed formulae (wff)

Pronounced "wiff"



Well-formed formula

From Wikipedia, the free encyclopedia

For the New Zealand government policy, see Working for Families.

In logic, **WFF** (pronounced "wiff") is an abbreviation for **well-formed formula**. Given a formal grammar, a WFF is any string that is generated by that grammar.

For example, in propositional logic the sequence of symbols $((\alpha \rightarrow \beta) \rightarrow (\neg\beta \rightarrow \neg\alpha))$ is a WFF because it is grammatically correct. The sequence of symbols $((\alpha \rightarrow \beta) \rightarrow (\beta\beta))\alpha))$ is not a WFF, because it does not conform to the grammar of propositional calculus.

In formal logic, proofs are sequences of WFFs with certain properties, and the final WFF in the sequence is what is proven. This is the basis for an esoteric pun used in the name of a product: "WFF 'n Proof: The Game of Modern Logic," by Layman Allen, a professor at the University of Michigan. The board game is designed to teach the principles of symbolic logic to children (in Polish notation), and its name is a pun on *whiffenpoof*, a nonsense word used as a cheer at Yale University made popular in *The Whiffenpoof Song*.

See also

- Formula (mathematical logic)

External links

- Well-Formed Formula for First Order Predicate Logic - includes a short Java quiz.
- Well-Formed Formula at ProvenMath

Category: Logic

wff

Well formed formulae for the predicate calculus can be defined using T and F, propositional variables, and the operators (not, or, and, implies, biconditional)

Basis Step: T, F, *and* s (s *is a proposional variable*) *are wff's*

Recursive Step: *if* E *and* F *are wff's then*

$$(\neg E), (E \wedge F), (E \vee F), (E \rightarrow F), \text{ and } (E \leftrightarrow F) \text{ are wff's}$$

fin