

# Recursion in Mathematics and Programming

# Summation

Simple examples

Evaluating  
expressions

Recursion in  
Mathematics

L-systems

Compute summation

$$\sum_{k=1}^n = 1 + 2 + \dots + n.$$

It's recursive definition:

$$\text{sum}(0) = 0$$

$$\text{sum}(n) = \text{sum}(n - 1) + n \quad (\text{for } n \geq 1)$$

Source code “*sum.jl*”.

# Summation

Simple examples

Evaluating  
expressions

Recursion in  
Mathematics

L-systems

Evaluation of the expression **sum**(4)

$$\begin{aligned}\text{sum}(4) &\Rightarrow \text{sum}(3) + 4 \\ &\Rightarrow (\text{sum}(2) + 3) + 4 \\ &\Rightarrow ((\text{sum}(1) + 2) + 3) + 4 \\ &\Rightarrow (((\text{sum}(0) + 1) + 2) + 3) + 4 \\ &\Rightarrow (((0 + 1) + 2) + 3 + 4) \\ &\Rightarrow ((1 + 2) + 3) + 4 \\ &\Rightarrow (3 + 3) + 4 \\ &\Rightarrow 6 + 4 \\ &\Rightarrow 10\end{aligned}$$

# Summation

Simple examples

Evaluating  
expressions

Recursion in  
Mathematics

L-systems

**sum**(4)

|

**sum**(3)

|

**sum**(2)

|

**sum**(1)

|

**sum**(0)

To compute **sum**(4), the function **sum** was called 5 times.

And to compute **sum**( $n$ ), the function **sum** will be called  $n+1$  times.

# Fibonacci numbers

Simple examples

Evaluating  
expressions

Recursion in  
Mathematics

L-systems

Recursive definition of the Fibonacci numbers:

$$\text{fib}(0) = 0$$

$$\text{fib}(1) = 1$$

$$\text{fib}(n) = \text{fib}(n-1) + \text{fib}(n-2) \quad (\text{for } n \geq 2)$$

Source code “*fib.jl*”.

# Fibonacci numbers

Simple examples

Evaluating  
expressions

Recursion in  
Mathematics

L-systems

Evaluation of the expression **fib**(5)

$$\begin{aligned}\mathbf{fib}(5) &\Rightarrow \mathbf{fib}(3) + \mathbf{fib}(4) \\&\Rightarrow (\mathbf{fib}(1) + \mathbf{fib}(2)) + (\mathbf{fib}(2) + \mathbf{fib}(3)) \\&\Rightarrow (1 + (\mathbf{fib}(0) + \mathbf{fib}(1))) + ((\mathbf{fib}(0) + \mathbf{fib}(1)) + (\mathbf{fib}(1) + \mathbf{fib}(2))) \\&\Rightarrow (1 + (0 + 1)) + ((0 + 1) + (1 + (\mathbf{fib}(0) + \mathbf{fib}(1)))) \\&\Rightarrow (1 + 1) + (1 + (1 + (0 + 1))) \\&\Rightarrow 2 + (1 + (1 + 1)) \\&\Rightarrow 2 + (1 + 2) \\&\Rightarrow 2 + 3 \\&\Rightarrow 5\end{aligned}$$

How many times did we call function **fib** to compute **fib**(5)?

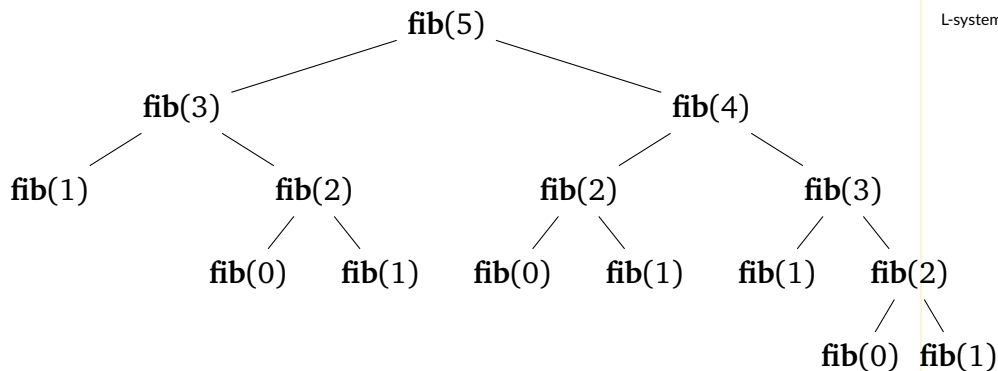
# Fibonacci numbers

Simple examples

Evaluating  
expressions

Recursion in  
Mathematics

L-systems



To compute `fib( $n$ )`, we need to do approximately  $2^n$  function calls.

Exponential running time,  $O(2^n)$ , too slow to be practical.

# Improved recursive Fibonacci

Simple examples

Evaluating  
expressions

Recursion in  
Mathematics

L-systems

Evaluation of the expression **fib\_loop\_rec**(5)

**fib\_loop\_rec**(5)     $\Rightarrow$     **loop**(1, 0, 1)  
                           $\Rightarrow$     **loop**(2, 1, 1)  
                           $\Rightarrow$     **loop**(3, 1, 2)  
                           $\Rightarrow$     **loop**(4, 2, 3)  
                           $\Rightarrow$     **loop**(5, 3, 5)  
                           $\Rightarrow$     5

Just  $n + 1$  function calls to compute **fib\_loop\_rec**( $n$ ).

This is a linear time  $O(n)$  algorithm, we get a real improvement!



# A simple calculator language

Simple examples

Evaluating  
expressions

Recursion in  
Mathematics

L-systems

We want to describe a computer that can do the following:

Input		Output
55	$\Rightarrow$	55
( 1 + 2 )	$\Rightarrow$	3
( 12 + ( ( 5 × 2 ) × 7 ) ) )	$\Rightarrow$	82

# A simple calculator language

Simple examples

Evaluating  
expressions

Recursion in  
Mathematics

L-systems

100

( 5 + ( 74 - 15 ) )

( ( 12 + 5 ) × ( 5 - 7 ) ) )

The syntax of the language:

$\langle expr \rangle ::=$  *strings of digits*, represent integer numbers

| (  $\langle expr \rangle$  +  $\langle expr \rangle$  )

| (  $\langle expr \rangle$  -  $\langle expr \rangle$  )

| (  $\langle expr \rangle$  ×  $\langle expr \rangle$  )

# A simple calculator language

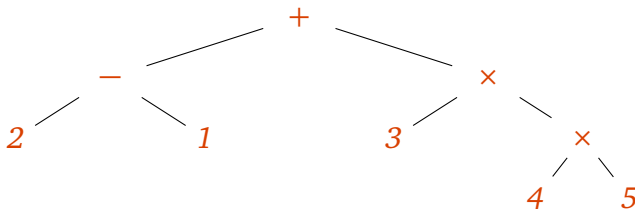
Simple examples

Evaluating  
expressions

Recursion in  
Mathematics

L-systems

$( ( 2 - 1 ) + ( 3 \times ( 4 \times 5 ) ) )$



The evaluating function looks at the root of the tree only and decides what to do

$$\begin{aligned} & eval[ ( E_1 + E_2 ) ] \\ \Rightarrow & eval[E_1] + eval[E_2] \end{aligned}$$

# A simple calculator language

Simple examples

Evaluating  
expressions

Recursion in  
Mathematics

L-systems

Evaluation function:

$$\text{eval}[(E_1 + E_2)] = \text{eval}[E_1] + \text{eval}[E_2]$$

$$\text{eval}[(E_1 - E_2)] = \text{eval}[E_1] - \text{eval}[E_2]$$

$$\text{eval}[(E_1 \times E_2)] = \text{eval}[E_1] \times \text{eval}[E_2]$$

$$\text{eval}[n] = n$$

# A simple calculator language

Simple examples

Evaluating  
expressions

Recursion in  
Mathematics

L-systems

$$\begin{aligned} & eval[ ( ( 2 - 1 ) + ( 3 \times ( 4 \times 5 ) ) ) ] \\ \Rightarrow & eval[ ( 2 - 1 ) ] + eval[ ( 3 \times ( 4 \times 5 ) ) ] \\ \Rightarrow & (eval[2] - eval[1]) + (eval[3] \times eval[ ( 4 \times 5 ) ]) \\ \Rightarrow & (2 - 1) + (3 \times (eval[4] \times eval[5])) \\ \Rightarrow & 1 + (3 \times (4 \times 5)) \\ \Rightarrow & 1 + (3 \times 20) \\ \Rightarrow & 1 + 60 \\ \Rightarrow & 61 \end{aligned}$$

# Recursion in Mathematics

Simple examples

Evaluating  
expressions

Recursion in  
Mathematics

L-systems

Some prominent examples of recursion

- *Euclid's algorithm*  
a algorithm for computing the greatest common divisor.
- *Newton's method*  
a recursive method for finding successively better approximations to the roots of a real-valued function.
- Complex objects such as fractals can be defined using recursive definition.

# Consider a recurrence

Simple examples

Evaluating  
expressions

Recursion in  
Mathematics

L-systems

Given a number  $c$ ,

$$z_0 = 0$$

$$z_n = z_{n-1}^2 + c$$

What sequences of numbers can be generated with this recurrence?  
The result, of course, depends on the value of  $c$ .

# Consider a recurrence

Simple examples

Evaluating  
expressions

Recursion in  
Mathematics

L-systems

$$z_0 = 0$$
$$z_n = z_{n-1}^2 + c$$

Let us test different values for  $c$ .

when  $c = 0$ :  $0 \rightarrow 0 \rightarrow 0 \rightarrow 0 \rightarrow 0 \rightarrow \dots$

when  $c = -1$ :  $0 \rightarrow (-1) \rightarrow 0 \rightarrow (-1) \rightarrow \dots$

when  $c = -2$ :  $0 \rightarrow (-2) \rightarrow 2 \rightarrow 2 \rightarrow 2 \rightarrow \dots$

when  $c = 1$ :  $0 \rightarrow 1 \rightarrow 2 \rightarrow 5 \rightarrow 26 \rightarrow \dots \infty$

when  $c = -3$ :  $0 \rightarrow (-3) \rightarrow 6 \rightarrow 33 \rightarrow \dots \infty$



# Define set $M$

when  $c = 0$ :  $0 \rightarrow 0 \rightarrow 0 \rightarrow 0 \rightarrow 0 \rightarrow \dots$

when  $c = -1$ :  $0 \rightarrow (-1) \rightarrow 0 \rightarrow (-1) \rightarrow \dots$

when  $c = -2$ :  $0 \rightarrow (-2) \rightarrow 2 \rightarrow 2 \rightarrow 2 \rightarrow \dots$

when  $c = 1$ :  $0 \rightarrow 1 \rightarrow 2 \rightarrow 5 \rightarrow 26 \rightarrow \dots \infty$

when  $c = -3$ :  $0 \rightarrow (-3) \rightarrow 6 \rightarrow 33 \rightarrow \dots \infty$

We are interested in all  $c$ , such that the corresponding sequence  $z_n$  **does not go to infinity**.

Let's say that all such numbers  $c$  belong to the set  $M$ .

Simple examples

Evaluating  
expressions

Recursion in  
Mathematics

L-systems

# Define set $M$

when  $c = 0$ :  $0 \rightarrow 0 \rightarrow 0 \rightarrow 0 \rightarrow 0 \rightarrow \dots$

when  $c = -1$ :  $0 \rightarrow (-1) \rightarrow 0 \rightarrow (-1) \rightarrow \dots$

when  $c = -2$ :  $0 \rightarrow (-2) \rightarrow 2 \rightarrow 2 \rightarrow 2 \rightarrow \dots$

when  $c = 1$ :  $0 \rightarrow 1 \rightarrow 2 \rightarrow 5 \rightarrow 26 \rightarrow \dots \infty$

when  $c = -3$ :  $0 \rightarrow (-3) \rightarrow 6 \rightarrow 33 \rightarrow \dots \infty$

We are interested in all  $c$ , such that the corresponding sequence  $z_n$  **does not go to infinity**.

Let's say that all such numbers  $c$  belong to the set  $M$ .

Numbers  $c$  and  $z_n$  are *complex numbers*. So, we need to quickly learn how to add and multiply them.

Simple examples

Evaluating  
expressions

Recursion in  
Mathematics

L-systems

# Quick intro to complex numbers

Simple examples

Evaluating  
expressions

Recursion in  
Mathematics

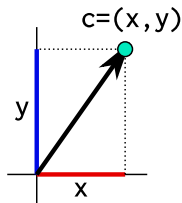
L-systems

The set of complex numbers  $\mathbb{C}$  is an extension of the set of real numbers  $\mathbb{R}$ .

Any complex number can be represented by a pair of real numbers

$$(x, y) \quad x, y \in \mathbb{R}$$

$x$  is the *real part*,  
 $y$  is the *imaginary part*.



*Alternative notation.* We can represent the pair as a sum of its real and imaginary part

$$(x, y) = x + yi$$

$i = \sqrt{-1}$  is the *imaginary unit*.

# Quick intro to complex numbers

Simple examples

Evaluating  
expressions

Recursion in  
Mathematics

L-systems

Addition

$$(a, b) + (c, d) = (a + c, b + d)$$

Multiplication

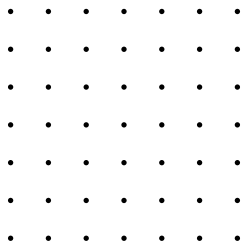
$$(a, b) \cdot (c, d) = (ac - bd, bc + ad)$$

Particularly,

$$i^2 = i \cdot i = (0, 1) \cdot (0, 1) = (0 - 1, 0 + 0) = (-1, 0) = -1$$

# How to draw the set $M$ ?

It's impossible to check all numbers  $c$ . But we can take a rectangular grid of points around the origin,  $(0,0)$ . We are going to mark all those points that belong to the set  $M$ .



$z_n = (x_n, y_n)$  will go to infinity if one of its components get large,  $\sqrt{x_n^2 + y_n^2} \geq 2$ . This condition is used practically to compute the membership of  $c$  in the set.

Source code “*mset.jl*”.

Simple examples

Evaluating  
expressions

Recursion in  
Mathematics

L-systems

# L-systems

An L-system or Lindenmayer system is a parallel *rewriting system*.

It consists of

- an alphabet of symbols
- an initial string (called an axiom) to start construction
- a collection of production rules that expand each symbol into some larger string of symbols

Example:

<i>Alphabet:</i>	$\{A, B\}$
<i>Initial string:</i>	$A$
<i>Production rules:</i>	$A \rightarrow AB$
	$B \rightarrow A$

Simple examples

Evaluating  
expressions

Recursion in  
Mathematics

L-systems

# L-systems

Example:

*Alphabet:*  $\{A, B\}$   
*Initial string:*  $A$   
*Production rules:*  $A \rightarrow AB$   
 $B \rightarrow A$

Start rewriting:

$n = 0 : A$

$n = 1 : AB$

$n = 2 : ABA$

$n = 3 : ABAAB$

$n = 4 : ABAABABA$

$n = 5 : ABAABABAABAAB$

$n = 6 : ABAABABAABAABAABAABABA$

Simple examples

Evaluating  
expressions

Recursion in  
Mathematics

L-systems

# L-systems

Koch curve:

*Alphabet:*  $\{F, L, R\}$   
*Initial string:*  $F$   
*Production rules:*  
 $F \rightarrow F L F R F R F L F$   
 $L \rightarrow L$   
 $R \rightarrow R$

When  $F$  stands for moving forward, and  $L$  and  $R$  are the commands for turning left and right



Source code “*lsys.jl*”.

Simple examples

Evaluating  
expressions

Recursion in  
Mathematics

L-systems