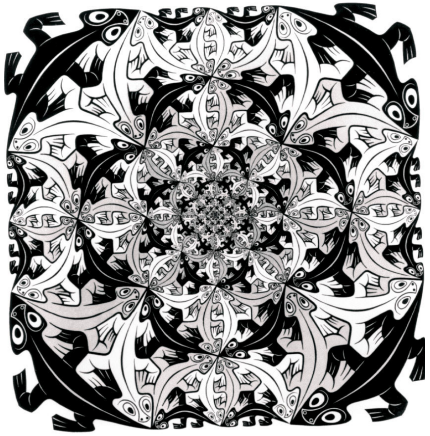
"Recursion" means to define something in terms of itself.

A directory is a collection of files and directories.

Words in dictionaries are defined in terms of other words.



"In order to understand recursion, one must first understand recursion."

– Anonymous

What is 83790 in base 8?

It's easy to find the last digit of a number `n` in base 8: It's simply `n % 8`.

The remaining digits are then the representation of `n / 8`.

```
def printBase8(n: Int) {
  if (n >= 8)
    printBase8(n / 8)
  print(n % 8)
}
```


We prove that `printBase8` is correct by induction on k, the number of digits of n in base 8.

Base Case: If $k = 1$, then $n < 8$, and `printBase8` prints one digit correctly.

Inductive Step: Let $k > 1$, so $n \geq 8$. We make the inductive assumption that `printBase8` works correctly for numbers with less than k digits. If we call `printBase8(n)`, then it recursively calls `printBase8(n/8)`. But $n/8$ has $k - 1$ digits in base 8, so this works correctly. Finally, the last digit is printed. It follows that `printBase8` prints n correctly.

Improved version

```
val DIGIT_TABLE = "0123456789abcdef"
val MAX_BASE    = DIGIT_TABLE.length

// Print n in any base, recursively
// Precondition: n >= 0, 2 <= base <= MAX_BASE
def printIntRec(n : Int, base : Int)
{
  if (n >= base)
    printIntRec(n / base, base)
  val digit = n % base
  print(DIGIT_TABLE(digit))
}
```


Correctness of recursive methods

When arguing about the correctness of a recursive method, always assume that the recursive call works.

Of course there has to be a base case.

And we need to be sure that we will reach the base case—there has to be some progress in each recursive call.

Factorial

Factorial: $n!$ is $n \times (n-1)!$.

```
// Evaluate n!
def factorial(n : Int) : Long = {
  if (n <= 1)     // base case
    1
  else
    n * factorial(n - 1)
}
```


Mistakes

Why doesn't this work?

```
def factorial(n : Int) : Long = {
  n * factorial(n - 1)
}
```

And this one?

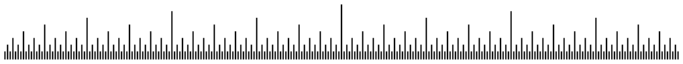
```
def factorial(n : Int) : Long = {
  if (n <= 1)     // base case
    1
  else
    n * factorial(n)
}
```

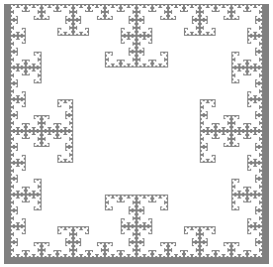
Recursive drawings

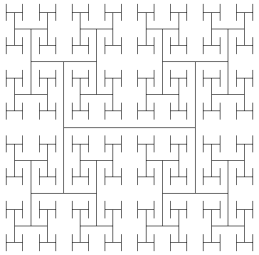
Ruler

Fractal star

H-Tree

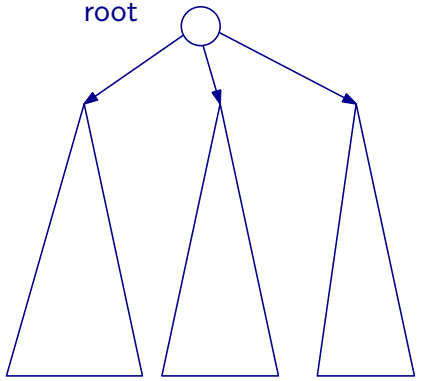


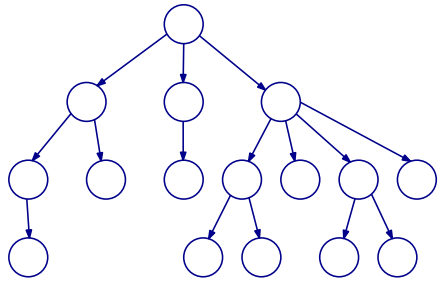




A recursive definition: trees

A tree consists of a root and zero or more subtrees, each of whose roots are connected to the root.





Each edge goes from the parent to the child.

Too much recursion

The Fibonacci numbers $F_0, F_1, F_2, \ldots$ are defined as follows: $F_0 = 0, F_1 = 1$, and $F_i = F_{i-1} + F_{i-2}$ for $i > 1$.

```
def fib(n : Int) : Long = {
  if (n == 0)      // base cases
    0
  else if (n == 1)
    1
  else
    fib(n - 1) + fib(n - 2)
}
```

Recursion is not useful when recursive calls duplicate work.
Don't solve the same subproblem in separate recursive calls.

Towers of Hanoi

Three poles, n discs.

One move: take the top disc from one pole and move it to another pole.

Goal: Move all discs from pole A to pole B.