# Recursive algorithms

An algorithm is called *recursive* if it solves a problem by reducing it to an instance of the same problem with smaller input.

Sometimes we can reduce the solution to a problem with a particular set of input values to the solution of the same problem with smaller input values.

The problem of finding the greatest common divisor of two positive integers $a$ and $b$, where $b > a$, can be reduced to gcd(b mod a, a) = gcd(a, b)

The solution to the original problem can be found with a sequence of reductions until the problem has been reduced to some initial case for which the solution is known

## Example 1

Give a recursive algorithm for computing the greatest common divisor of two nonnegative integers $a$ and $b$ with $a < b$.

**A Recursive Algorithm for Computing gcd(*a, b*).**

**procedure** $gcd(a, b$: nonnegative integers with $a < b)$
**if** $a = 0$ **then return** $b$
**else return** $gcd(b$ **mod** $a, a)$
{output is gcd$(a, b)$}

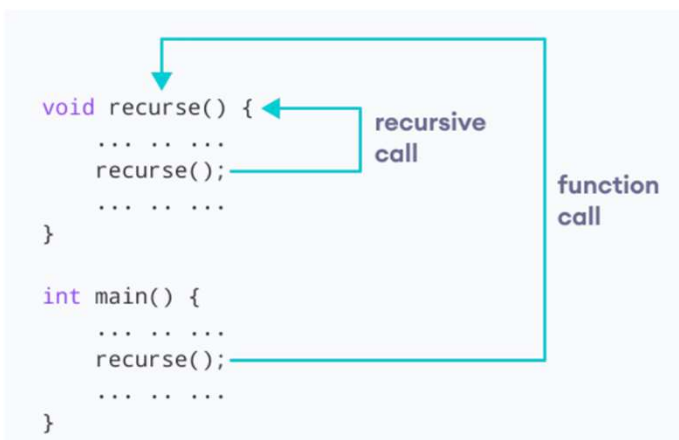
We illustrate the workings of Algorithm Inputs a = 5, b = 8.

# Example 2

Give a recursive algorithm for computing $n!$, where $n$ is a nonnegative integer

## A Recursive Algorithm for Computing $n!$.

**procedure** $factorial(n$: nonnegative integer)
**if** $n = 0$ **then return** 1
**else return** $n \cdot factorial(n-1)$
{output is $n!$}




return 5 * factorial(4) = 120
  return 4 * factorial(3) = 24
    return 3 * factorial(2) = 6
      return 2 * factorial(1) = 2
        return 1 * factorial(0) = 1

1 * 2 * 3 * 4 * 5 = 120

## Example 3

Give a recursive algorithm for computing $a^n$, where $a$ is a nonzero real number and $n$ is a nonnegative integer.

We can base a recursive algorithm on the recursive definition of $a^n$. This definition states that $a^{n+1} = a \cdot a^n$ for $n > 0$ and the initial condition $a^0 = 1$.

### A Recursive Algorithm for Computing $a^n$

**procedure** *power*($a$: nonzero real number, $n$: nonnegative integer)
**if** $n = 0$ **then return** 1
**else return** $a \cdot power(a, n - 1)$
{output is $a^n$}

## Example 3

Devise a recursive algorithm for computing $b^n$ **mod** $m$, where $b$, $n$, and $m$ are integers with $m \geq 2, n \geq 0,$ and $1 \leq b < m$.

*Solution:* We can base a recursive algorithm on the fact that

the initial condition $b^0 \textbf{ mod } m = 1$

$$b^n \textbf{ mod } m = (b \cdot (b^{n-1} \textbf{ mod } m)) \textbf{ mod } m,$$

we can devise a much more efficient recursive algorithm

$b^n \textbf{ mod } m = (b^{n/2} \textbf{ mod } m)^2 \textbf{ mod } m$ **When n is even**

$b^n \textbf{ mod } m = \left((b^{\lfloor n/2 \rfloor} \textbf{ mod } m)^2 \textbf{ mod } m \cdot b \textbf{ mod } m\right) \textbf{ mod } m$ **When n is odd**

## Example 3

Devise a recursive algorithm for computing $b^n$ **mod** $m$, where $b$, $n$, and $m$ are integers with $m \geq 2, n \geq 0,$ and $1 \leq b < m$.

$b^n \mathbf{\ mod\ } m = (b^{n/2} \mathbf{\ mod\ } m)^2 \mathbf{\ mod\ } m$ **When n is even**

the initial condition $b^0 \mathbf{\ mod\ } m = 1$

$b^n \mathbf{\ mod\ } m = \left((b^{\lfloor n/2 \rfloor} \mathbf{\ mod\ } m)^2 \mathbf{\ mod\ } m \cdot b \mathbf{\ mod\ } m\right) \mathbf{\ mod\ } m$ **When n is odd**

input $b = 2$, $n = 5$, and $m = 3$

First, $n = 5$ is odd: $mpower(2, 5, 3) = (mpower(2, 2, 3)^2 \mathbf{\ mod\ } 3 \cdot 2 \mathbf{\ mod\ } 3) \mathbf{\ mod\ } 3.$

*Second, n = 2 is even:* $mpower(2, 2, 3) = mpower(2, 1, 3)^2 \mathbf{\ mod\ } 3.$

*Third, n=1 is odd:* $mpower(2, 1, 3) = (mpower(2, 0, 3)^2 \mathbf{\ mod\ } 3 \cdot 2 \mathbf{\ mod\ } 3) \mathbf{\ mod\ } 3.$

*Fourth, n=0 is base case:* $mpower(2, 0, 3) = 1$

**Finally Working backwards**

## Example 3 continued

Devise a recursive algorithm for computing $b^n \textbf{ mod } m$, where $b$, $n$, and $m$ are integers with $m \geq 2, n \geq 0$, and $1 \leq b < m$.

$b^n \textbf{ mod } m = (b^{n/2} \textbf{ mod } m)^2 \textbf{ mod } m$ **When n is even**

the initial condition $b^0 \textbf{ mod } m = 1$

$b^n \textbf{ mod } m = \left((b^{\lfloor n/2 \rfloor} \textbf{ mod } m)^2 \textbf{ mod } m \cdot b \textbf{ mod } m\right) \textbf{ mod } m$ **When n is odd**

**ALGORITHM Recursive Modular Exponentiation.**

**procedure** *mpower*($b, n, m$: integers with $b > 0$ and $m \geq 2, n \geq 0$)
**if** $n = 0$ **then**
    **return** 1
**else if** $n$ is even **then**
    **return** $mpower(b, n/2, m)^2$ **mod** $m$
**else**
    **return** $(mpower(b, \lfloor n/2 \rfloor, m)^2 \textbf{ mod } m \cdot b \textbf{ mod } m) \textbf{ mod } m$
{output is $b^n$ **mod** $m$}