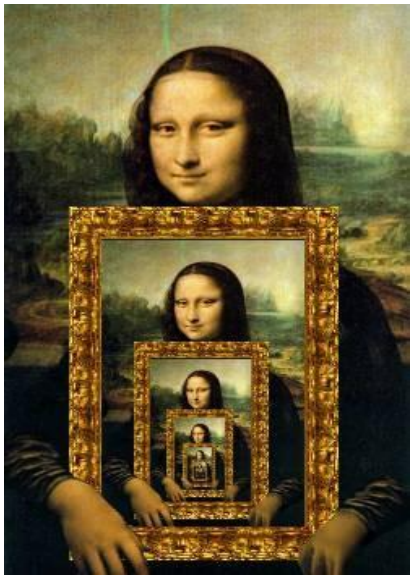
UNIVERSIDAD NACIONAL DE ASUNCIÓN
FACULTAD DE INGENIERÍA

Cursos Básicos
Primer Ciclo 2024

Fundamentos de Programación

Recursividad



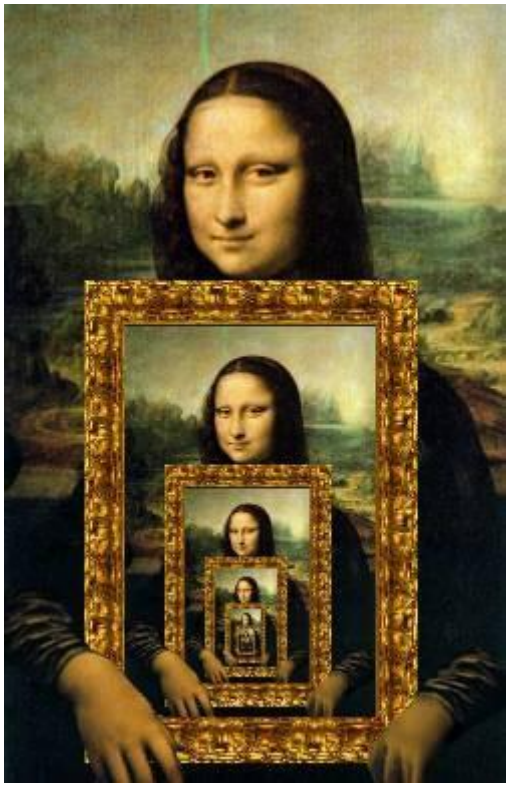
```
def factorial(n):
  if (n==1) :
     return 1
  else:
     return n*factorial(n-1)
```

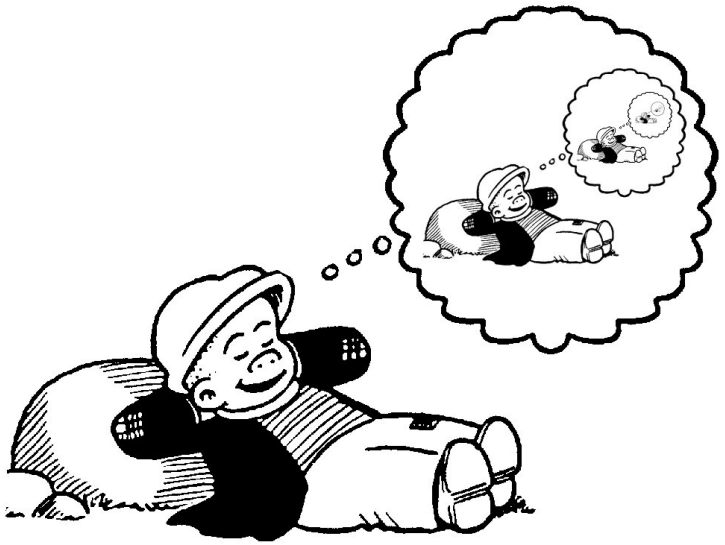
¿Qué veremos hoy?

- Introducción a la Recursividad

Recursividad

La *recursividad* (o *recurrencia*, o *recursión*) es una de las ideas centrales en ciencias de la computación. Resolver un problema mediante recursión significa que la solución depende de las soluciones de **casos más pequeños del mismo problema**.





Algoritmos recursivos

Un *algoritmo recursivo* es un algoritmo que expresa la solución de un problema en términos de una llamada a sí mismo. La llamada a sí mismo se conoce como llamada *recursiva* o *recurrente*.

Ejemplo: factorial de un número natural N

$$N! = 1 * 2 * 3 * \cdots * (N-1) * N$$

$$N! = \left(1 * 2 * 3 * \cdots * (N-1)\right) * N$$

$$N! = \boxed{(N-1)!} * N$$

Llamada recursiva

Recursividad

Una función que se llama a sí misma se denomina recursiva.

En la recursividad tenemos estos factores importantes:

a) Una función se llama a sí misma

b) En cada llamada, tiene que haber una condición que analiza el argumento de entrada (si no estaríamos en una autollamada tipo bucle infinito)

c) Tiene que existir una situación en la que ya no hay una nueva autollamada, sino un resultado o acción. Esta situación se denomina caso base y permite que se salga de la recursión.

Características

Los algoritmos recursivos tienen estas características:

1. La función debe llamarse a sí misma*.

Factorial de un número

$$N! = \boxed{(N-1)!} * N$$

Caso más pequeño del mismo problema

2. Tiene que existir una situación en la que ya no hay una nueva autollamada, sino un resultado o acción (o sino el proceso sería infinito). Esta situación se denomina **caso base** y permite que se salga de la recursión.

Factorial de un número

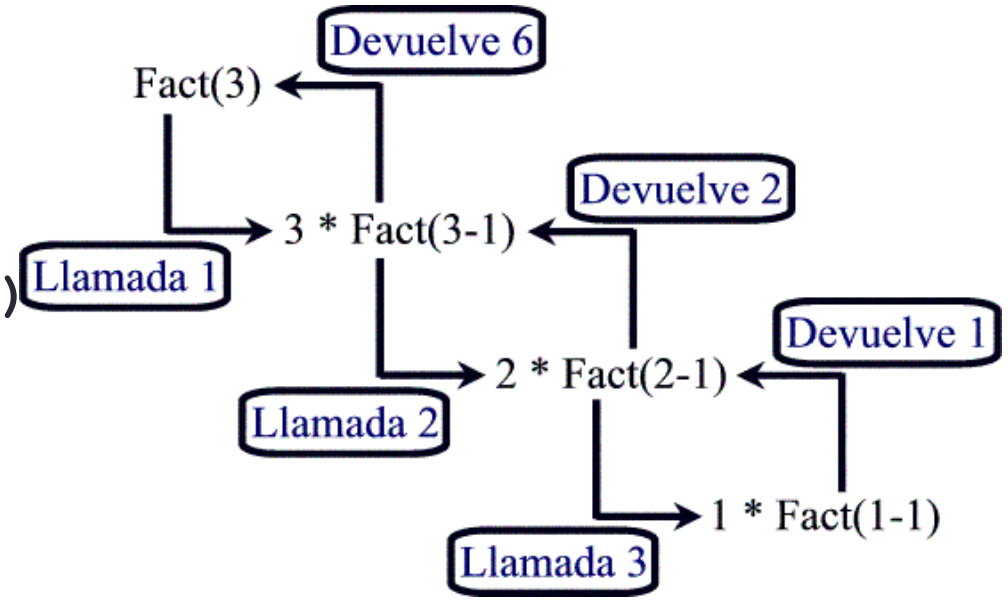
Caso base $\boxed{1! = 1}$

*Existe también la recursividad indirecta, pero no será tratado en el curso.

Recursividad

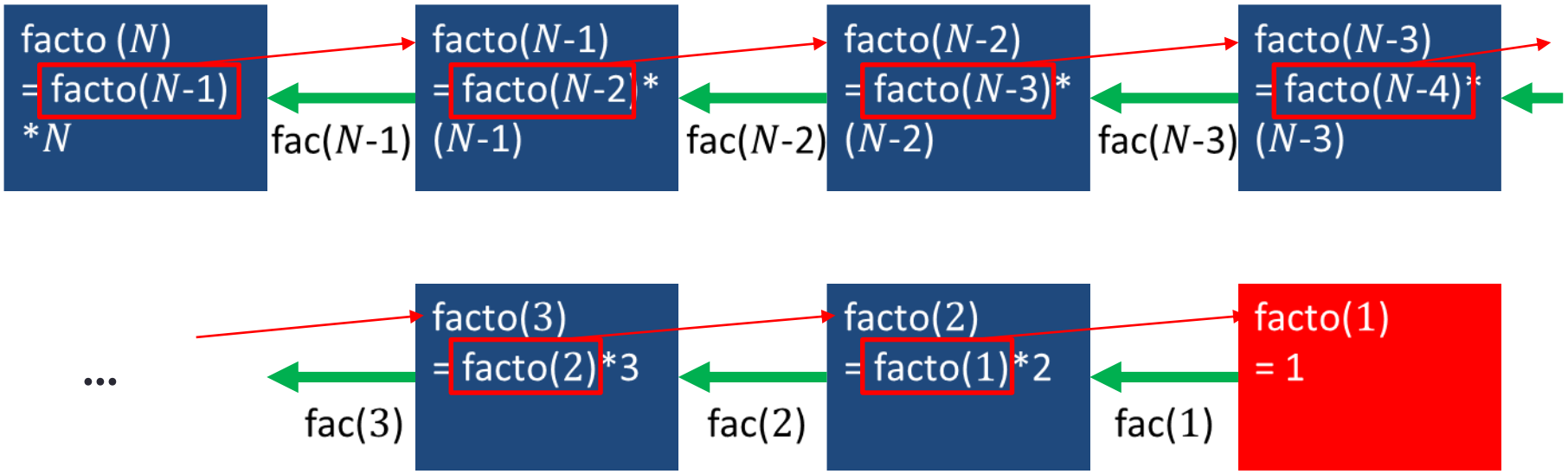
Calcular el factorial de 3

```
def factorial(n):
  if (n==1) :
     return 1
  else:
     return n*factorial(n-1)
```



Recursividad - Funcionamiento

Ejemplo: factorial de un número natural N – Función **facto**



Recursividad - Factorial

$$N! = (N-1)! * N$$

Caso base:

$$1! = 1$$

```
def factorial(n):
  if (n==1) :
     return 1
  else:
     return n*factorial(n-1)
```

Recursiones vs bucles

Recursión

```
def factorial(n):
  if (n==1) :
     return 1
  else:
     return n*factorial(n-1)
```

Bucle

```
def factorial(n):
    fac=1;
    for i in range(1,n+1):
      fac*=i
     return fac
```

Recursividad

Una función que se llama a sí misma se denomina recursiva.

Recursividad vs. Iteración

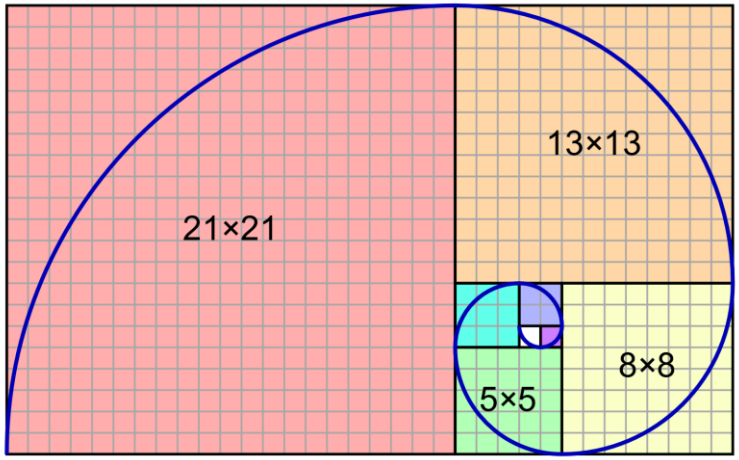
Aspectos que hay que considerar al decidir cómo implementar la solución a un problema (de forma iterativa o de forma recursiva)

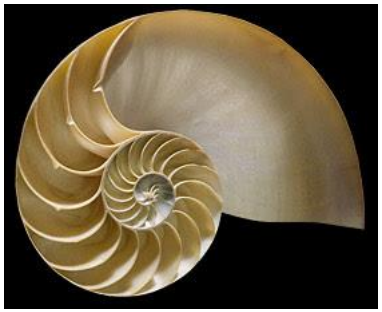
- La carga computacional (tiempo de CPU y espacio en memoria) asociada a las llamadas recursivas.
-La redundancia (algunas soluciones recursivas resuelven un problema en repetidas ocasiones).
- La complejidad de la solución (en ocasiones, la solución iterativa es muy difícil de encontrar).
- La concisión, legibilidad y elegancia del código resultante de la solución recursiva del problema

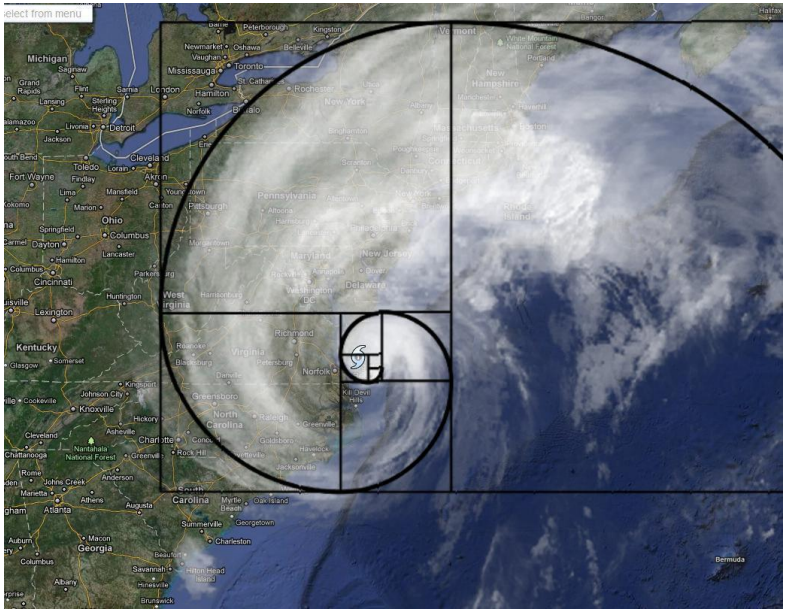
Ejemplos

Ejemplo 3: Calcular el enésimo término de la sucesión de Fibonacci, donde cada término es la suma de los dos anteriores, y sabiendo que los dos primeros términos son 0 y 1.

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89…







Recursividad

Series de fibonacci

```
def fibonacci (n):
    ant1 = ant2 = 1;
    if ((n == 0) or (n == 1)):
       actual = 1;
    else:
        for i in range(2,n+1) :
            actual = ant1 + ant2
            ant2 = ant1
            ant1 = actual
         return (actual)
```

Recursividad

Series de fibonacci

```
def fibonacci (n) :
      if (n == 1) :
          return 0;
      elif (n == 2) :
          return 1;
      else:
          return fibonacci(n-1) +fibonacci(n-2)
```

¿Por qué recursiones?

La razón fundamental es que existen numerosos problemas complejos que poseen naturaleza recursiva y por ello son más fáciles de solucionar con algoritmos de este tipo.

Ejemplo:

Torres de Hanoi

¿Por qué recursiones?

Torres de Hanoi

Los discos se apilan sobre una varilla en tamaño decreciente. No hay dos discos iguales, y todos ellos están apilados de mayor a menor radio en una de las varillas, quedando las otras dos varillas vacantes.

Objetivo: El juego consiste en pasar todos los discos de la varilla ocupada (es decir la que posee la torre) a una de las otras varillas vacantes.

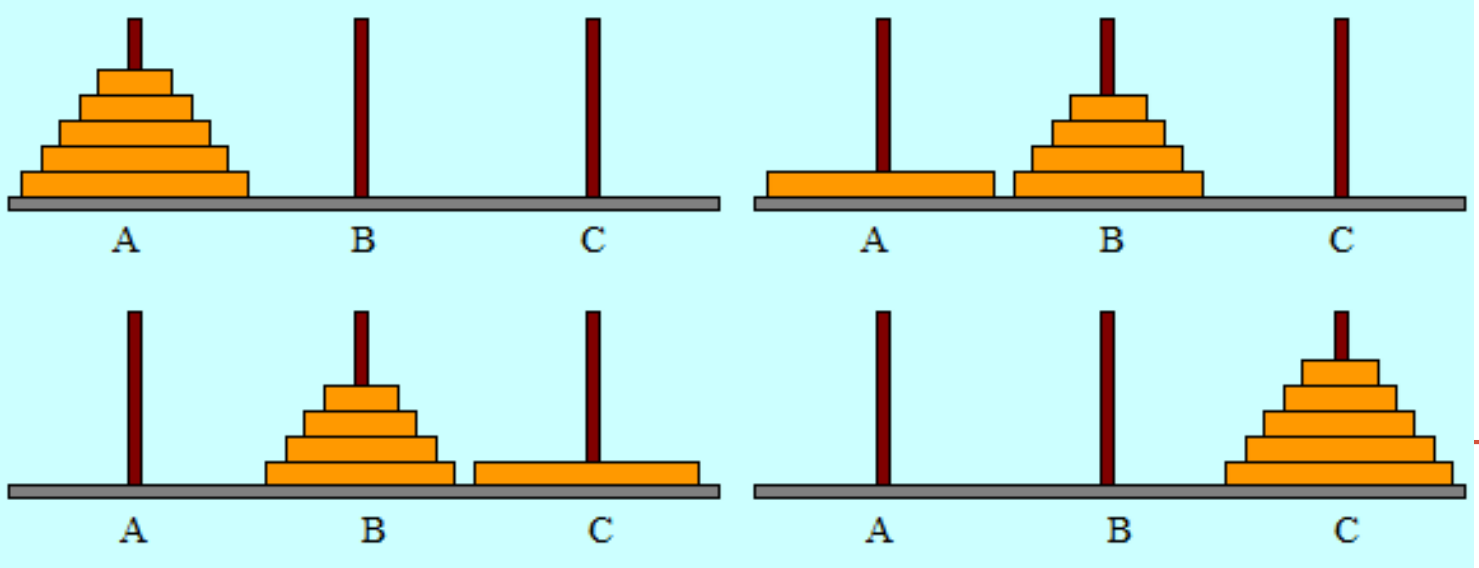
¿Por qué recursiones?

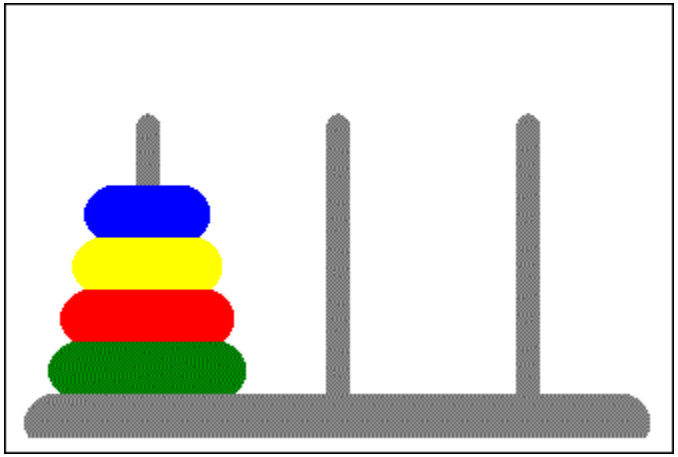
Torres de Hanoi

Reglas:

- Sólo se puede mover un disco cada vez.
- Un disco de mayor tamaño no puede descansar sobre uno más pequeño que él mismo.
- Sólo puedes desplazar el disco que se encuentre arriba en cada varilla.

¿Por qué recursiones?





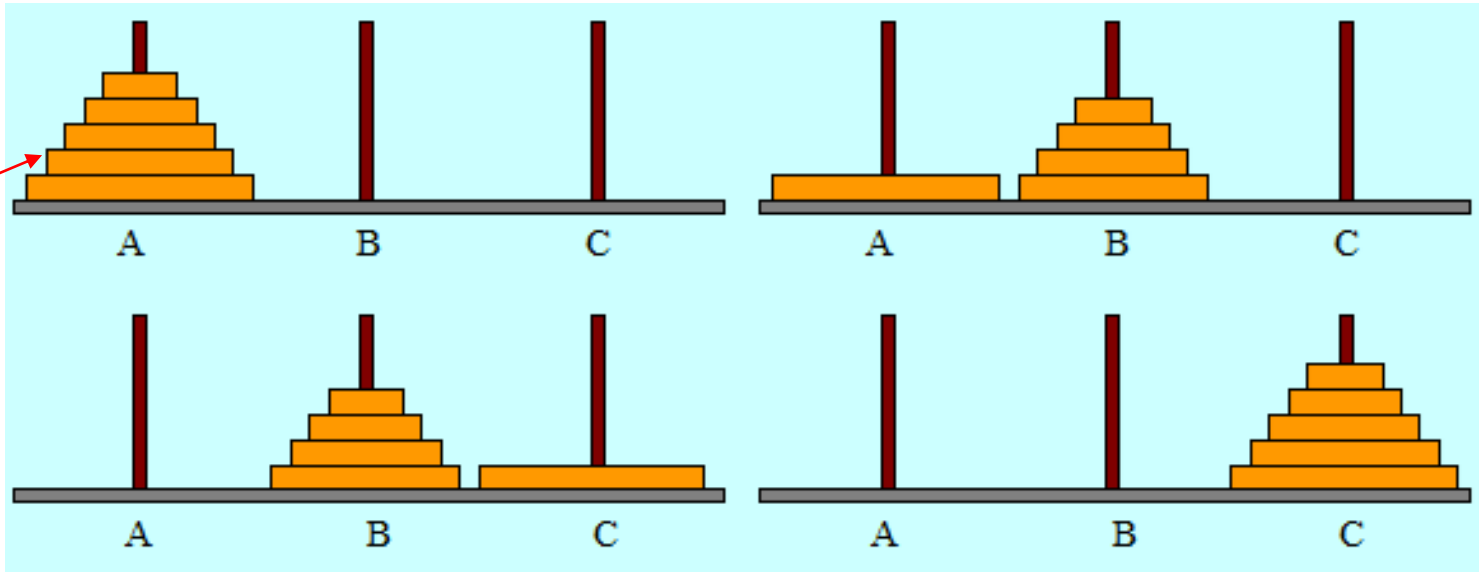
Solución – Torres de Hanoi

```
def Hanoi(inicio,fin, centro, n):
    if(n==1) :
        print("Mover disco 1 de la varilla “,inicio," a la varilla “,fin);
     else:
        Hanoi(inicio, centro, fin, (n-1));
        print("Mover disco ",n," de la varilla ",inicio," a la varilla “,fin)
        Hanoi(centro, fin, inicio, (n-1));
```

Hanoi(1,2,3,n)

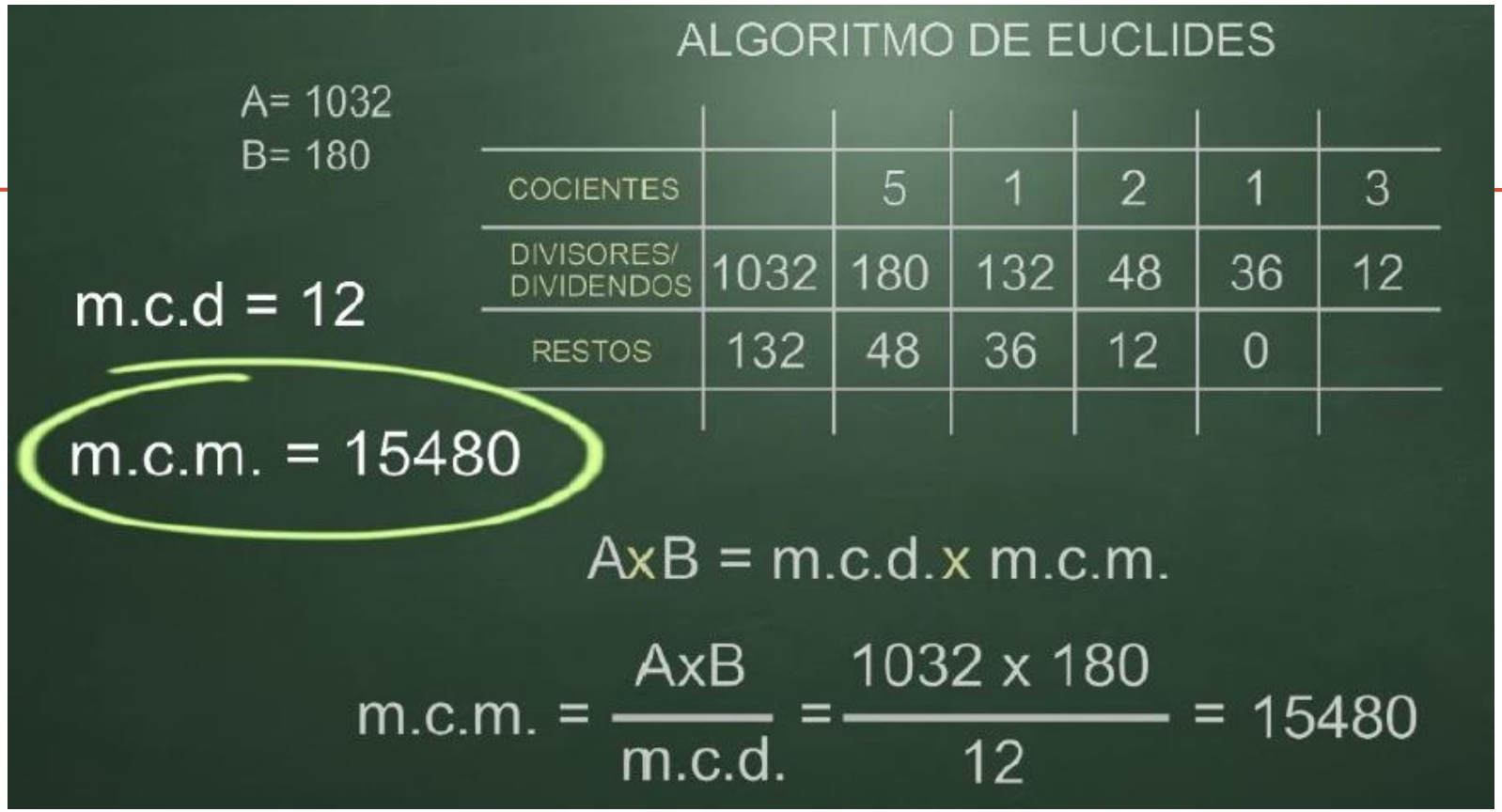
1 2 3

n discos



Ejercicio Propuesto

Ejercicio 1: Escribir un programa que muestre el máximo común divisor (MCD) de dos números A y B (usando el algoritmo de Euclides).



Ejercicios

Ejercicio 2 (Opcional): CompuPesos (problema extraído de ICPC/ACM y adaptado para el curso)

La república de Compu tiene un sistema monetario bastante particular. Cada moneda tiene escrito un número entero que indica su valor en Compupesos. Los bancos (aquí viene lo peculiar) permiten cambiar una moneda por otras tres de valores $n/2$, $n/3$ y $n/4$; redondeadas hacia abajo.

Además, se tiene que un Compupeso equivale a un dólar estadounidense. Entonces, teniendo una moneda de un cierto valor X, ¿cuál es la mayor cantidad de dólares que pueden obtenerse de él?