Estructuras de datos (TC1018)

Tema 2. Recursión

Instituto Tecnológico y de Estudios Superiores de Monterrey. Campus Santa Fe
Departamento de Tecnologías de Información y Electrónica
Dr. Vicente Cubells (vcubells@itesm.mx)

Temario

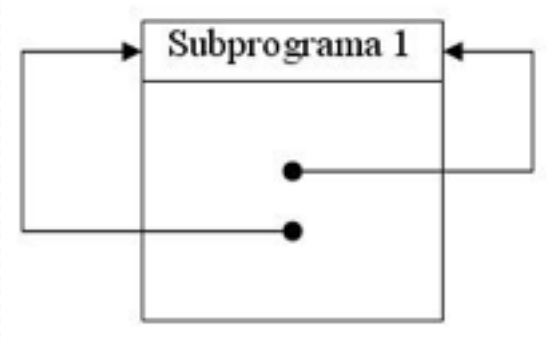
- Análisis de algoritmos recursivos
  - Definición de recursividad
  - Ejemplos de algoritmos recursivos

Ejemplo Matrushka

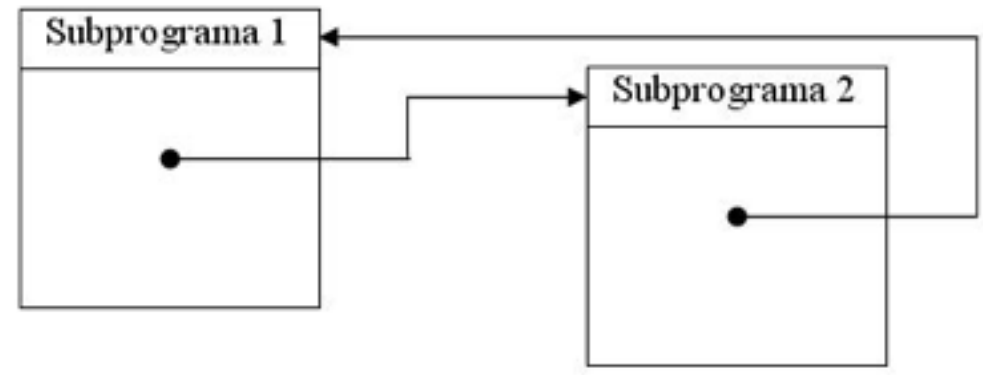
- La Matrushka es una artesanía tradicional rusa. Es una muñeca de madera que contiene otra muñeca más pequeña dentro de sí. Esta muñeca, también contiene otra muñeca adentro. Y así, una adentro de la otra.

Recursividad…

- Un algoritmo es recursivo si se invoca a si mismo
- Tipos
  - Directa (simple o múltiple)



  - Indirecta

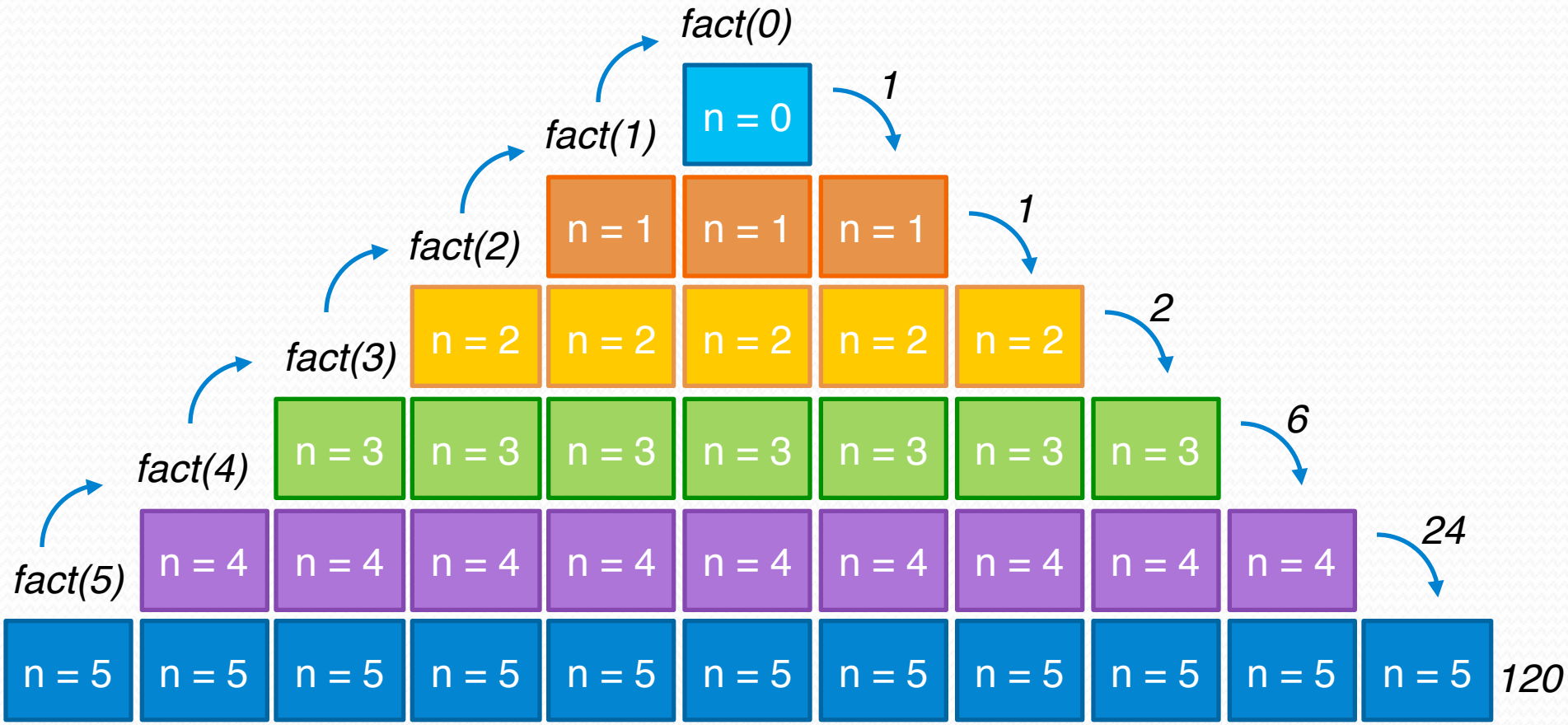


Recursividad…

- Características de una función recursiva:
  - Dos pasos:
    - paso base o condición de parada
    - paso recursivo
  - En cada ciclo, debe ir acercándose más a la solución
    - Ejemplos:
      - Factorial
      - Potencia n
      - Serie de Fibonacci
      - Método de Ackermann
      - Entre otros

Factorial

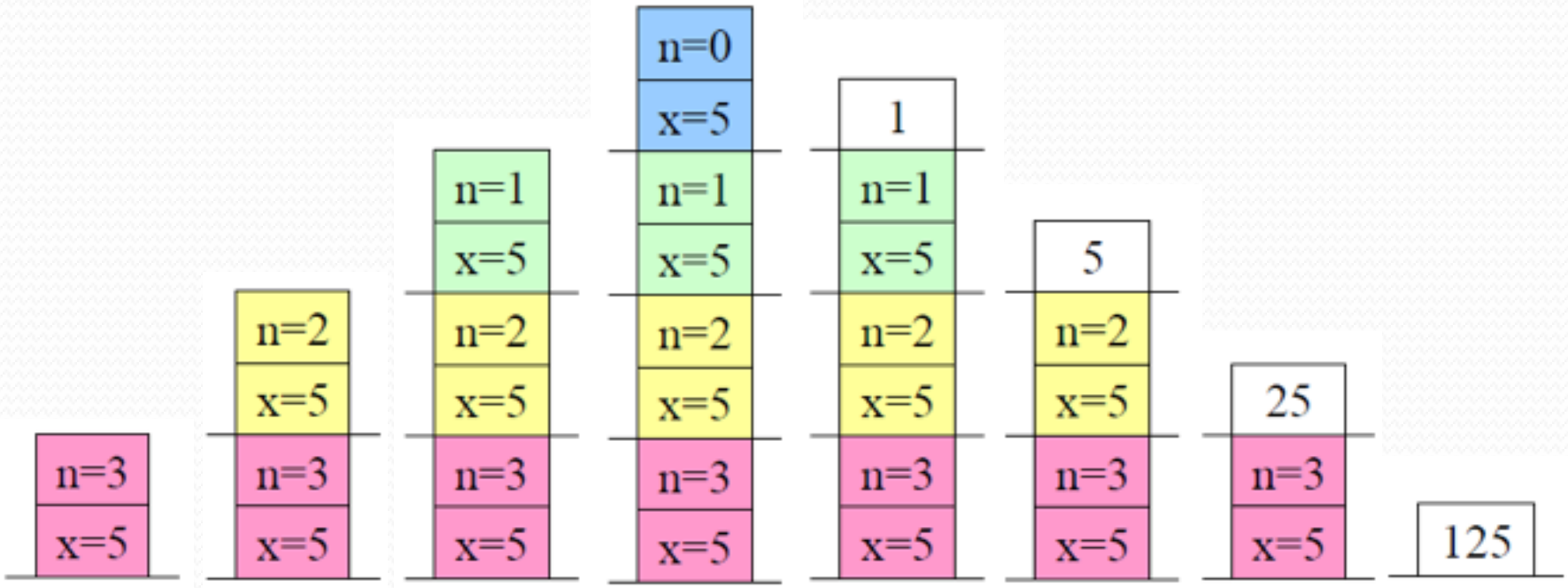
$$n! = \begin{cases} 1 & \text{si } n = 0 \\ n \cdot (n-1)! & \text{si } n > 0 \end{cases}$$



Potencia *n*

$$x^n = \begin{cases} 1 & \text{si } n = 0 \\ x \cdot x^{n-1} & \text{si } n > 0 \end{cases}$$

Cálculo de 5^3



Otras funciones recursivas

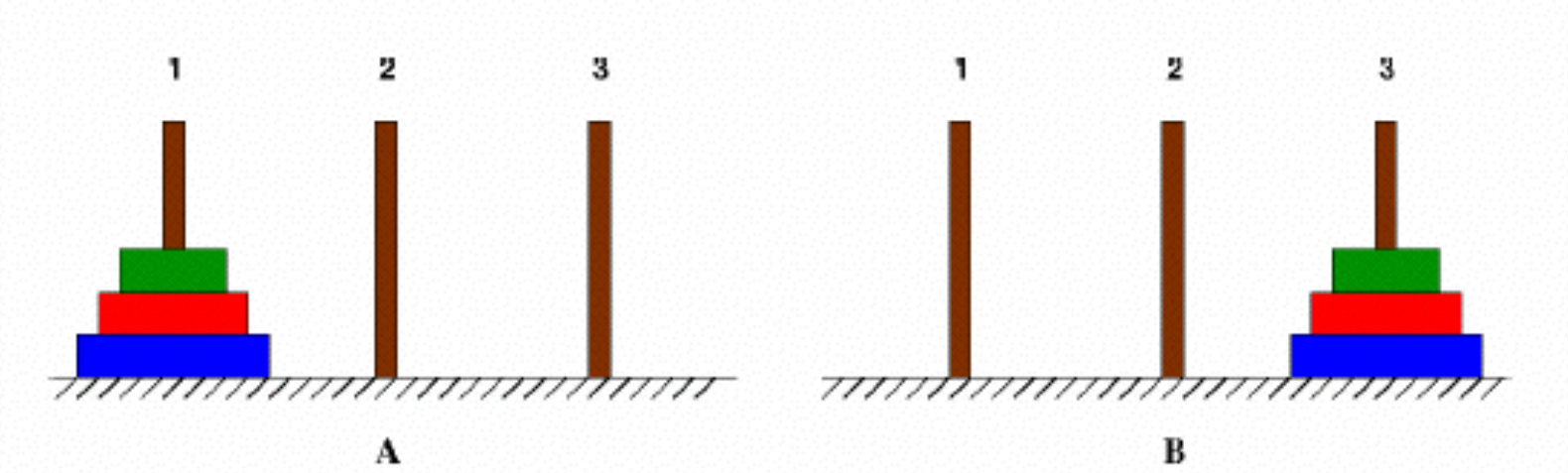
- Serie de Fibonacci

$$f_i = \begin{cases} 0 & \text{si } i = 0 \\ 1 & \text{si } i = 1 \\ f_{(i-2)} + f_{(i-1)} & \text{si } i > 1 \end{cases}$$

- Función de Ackermann

$$A(m,n) = \begin{cases} n+1, & \text{si } m = 0; \\ A(m-1,1), & \text{si } m > 0 \text{ y } n = 0; \\ A(m-1, A(m,n-1)), & \text{si } m > 0 \text{ y } n > 0 \end{cases}$$

Las Torres de Hanoi



Mover n discos del poste 1 al poste 3 (utilizando el poste 2 como auxiliar): hanoi (n, 1, 2, 3)

Las Torres de Hanoi

```
void hanoi( n, x, y, z )
{
   if (n == 1)
     move ( x, y );
   else
   {
     hanoi( n-1, x, z, y );
     move ( x, y );
     hanoi( n-1, z, y, x );
   }
}
```

Recursividad

- Ventajas
  - Solución natural
  - Resuelve problemas complejos
- Desventajas
  - Se puede llegar a un ciclo infinito
  - Solución iterativa más difícil de implementar
  - Es difícil de programar

Resumiendo

- La recursividad puede resolver problemas complejos de manera simple
- Un algoritmo recursivo siempre debe tener una condición de parada
- Un algoritmo iterativo siempre será más eficiente que su contraparte recursiva