Tema 2 Programación Recursiva

Estrategias de Programación y Estructuras de Datos

CA Guadalajara (UNED)

Programación Recursiva

En primer lugar, se debe presentar el concepto de **recursión**. La **recursión** es un método matemático que se llama **a sí mismo**, en un contexto donde cada llamada es **más simple** que la anterior. Por tanto:

- **(Caso base)**: Es necesario que exista un caso base, que se resuelva sin necesidad de usar la recursividad.
- **(Progresión)**: Cada llamada recursiva debe ir progresando hacia el caso base.

Programación Recursiva

Para definir una **función/método recursivo**, se deben identificar los siguientes parámetros:

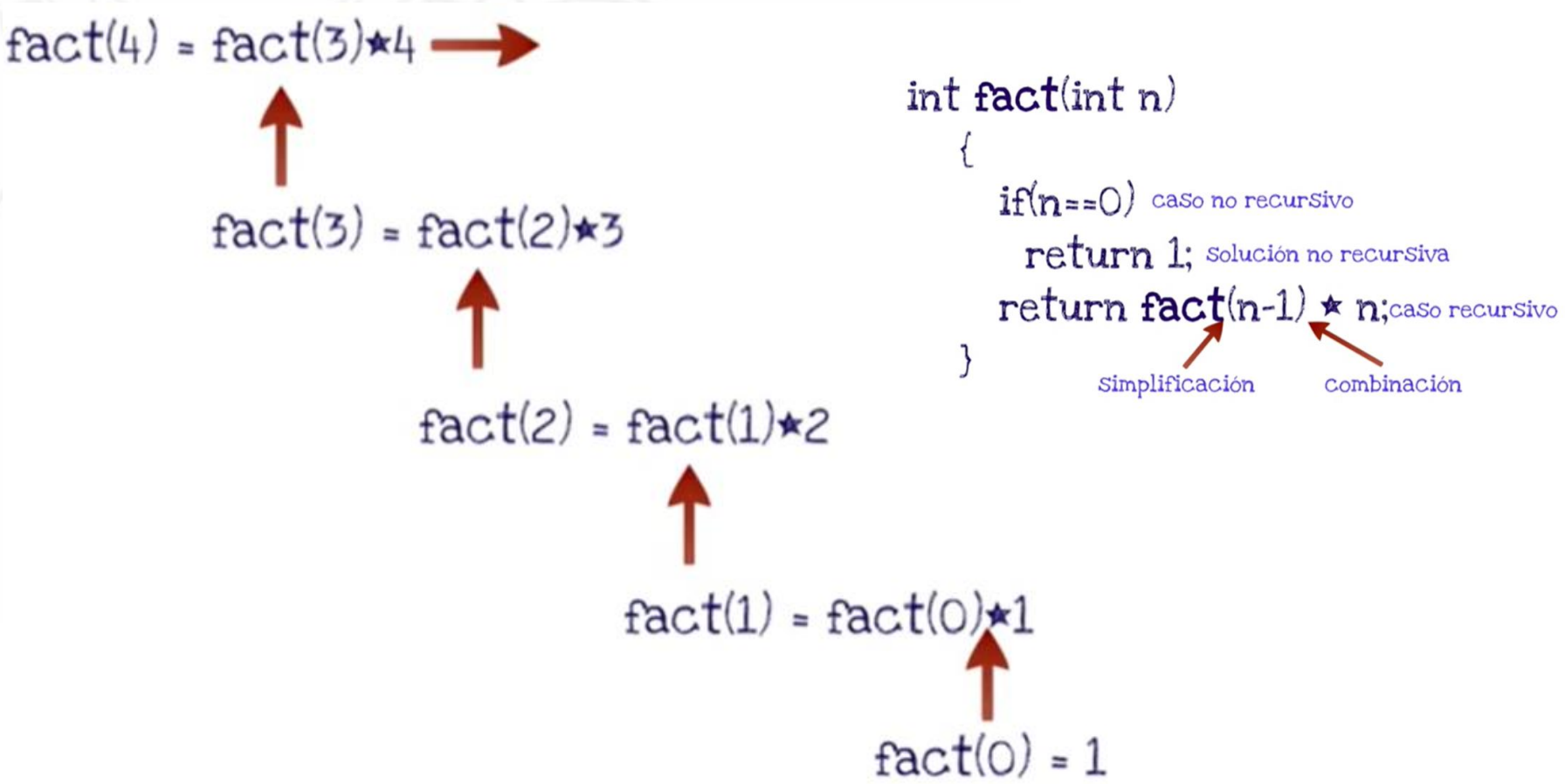
1. Identificar la función recursiva.
2. Identificar el caso base (no reducible y por tanto, su resultado)
3. Identificar la función para combinar/reducir la complejidad en las llamadas.

Lo vemos en un par de ejemplos. El primero el factorial:

1. Fact(n) = n * Fact(n-1)
2. Si n=0 -> 0! = 1
3. Si n>0 -> n! = n * Fact (n-1)

```
public static long factorial(int n){
if(n <= 1) // caso base
        return 1;
else
        return n * factorial (n - 1);
}
```

Programación Recursiva



Programación Recursiva

El segundo el cálculo de término e-nésimo de la **serie de Fibonacci**:

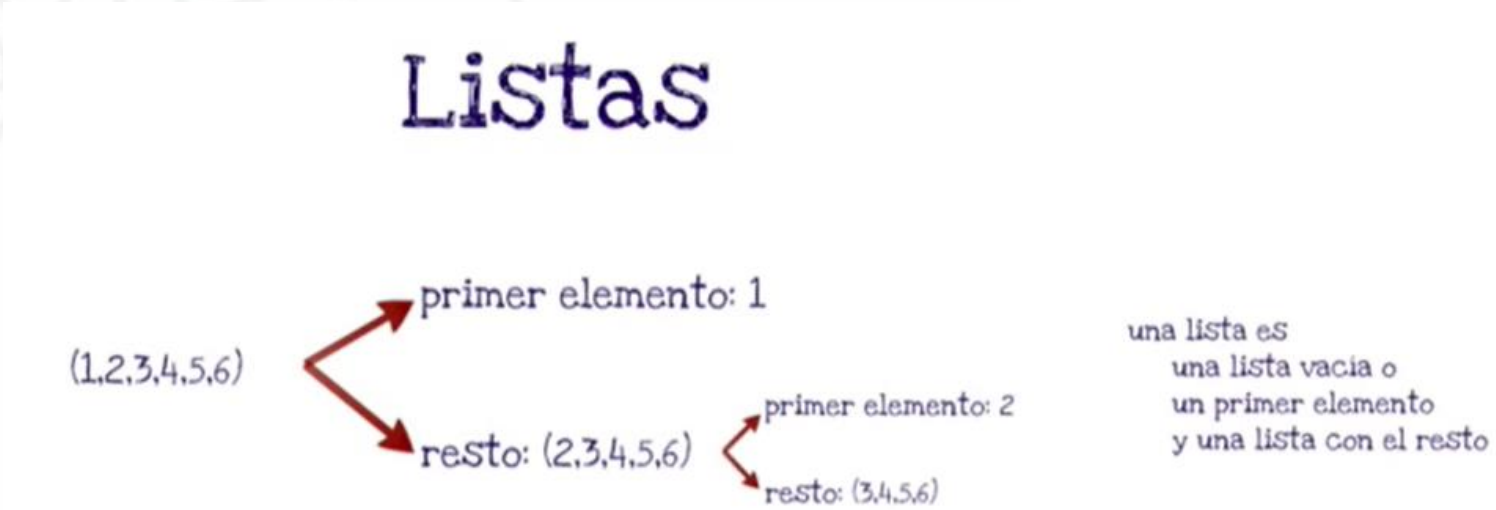
1. F(n) = F(n-1) + F(n-2)
2. Si n=0 -> F(0) = 0 y Si n=1 -> F(1)=1
3. Si n>0 -> F(n) = F(n-1) + F(n-2)

```
int fibonacci(int n) {

if (n>1){
     return fibonacci(n-1) + fibonacci(n-2); //función recursiva
 }
 else if (n==1) { // caso base
     return 1;
 }
 else if (n==0){ // caso base
     return 0;
 }
 else{ //error
    System.out.println("El tamaño debe ser mayor o igual a 1");
    return -1;
   }
}
```

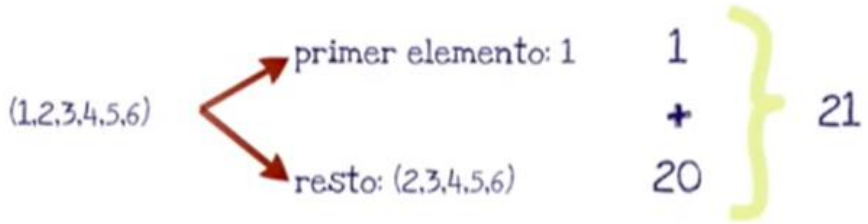
Programación Recursiva

Para una Estructura de Datos básica como las Listas:



Programación Recursiva

Suma de los elementos de una lista



caso no recursivo

Suma(lista_vacía) = 0

caso recursivo

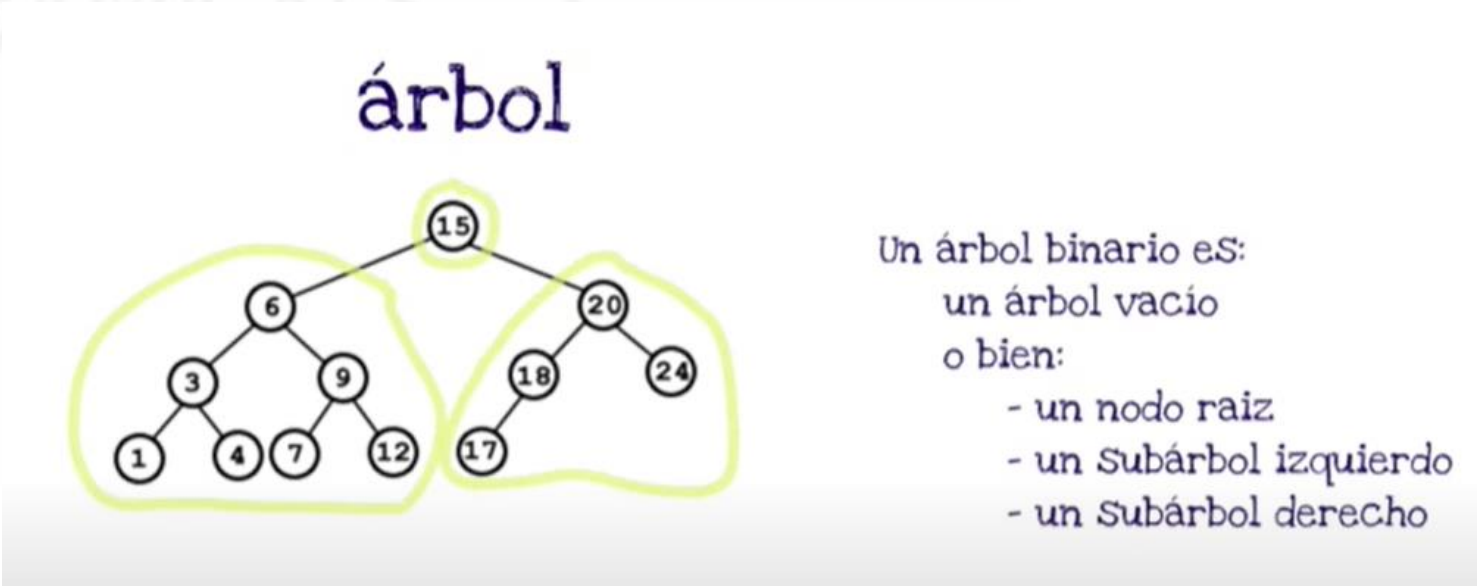
Suma(lista) = primero(lista) + **Suma**(resto(lista))

combinación

simplificación

Programación Recursiva

Para una Estructura de Datos Básica como las Árbol:

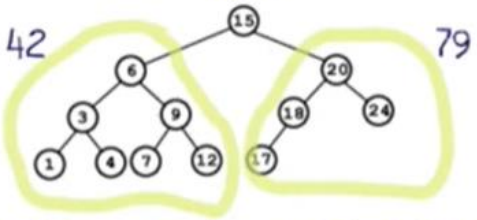


Programación Recursiva

Suma de los elementos de un árbol

raiz

42



79

Subárbol izquierdo Subárbol derecho

Suma(árbol) =
15 + 42 + 79

caso no recursivo

Suma(árbol_vacío) = 0

caso recursivo

Suma(árbol) = valor de la raiz
+ **Suma**(árbol_izquierdo)
+ **Suma**(árbol_derecho)

Programación Recursiva

Existen varios tipos de funciones recursivas en función del número de llamadas recursivas y según el método de disminución de los datos.

- **Según el número de llamadas:** Existen funciones recursivas **lineales**, con un único sucesor y funciones **recursivas múltiples** donde existen varias llamadas recursivas

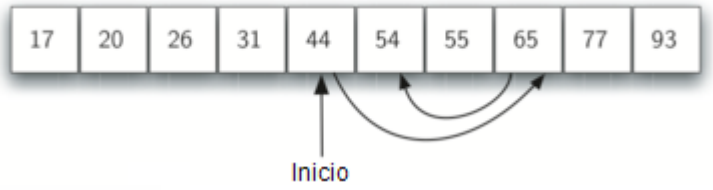
- **Según el métodos de disminución**: Mediante mecanismos de tipo sustracción (resta) y división.

Por ejemplo, la función recursiva del factorial es **lineal-sustracción**, la serie Fibonacci es **múltiple-sustracción**

Programación Recursiva

Para el caso de **lineal-división**, lo veremos con un ejemplo práctico de búsqueda dicotómica, asumiendo que el espacio de números está ordenado. La idea es buscar en una lista de números, por ejemplo una array. Se identifica el primero con el que comparar y si es mayor o menor se toma la decisión para el siguiente paso…. Así sucesivamente. Donde:

```
int busqueda_binaria (int A[ ],int X, int i, int j)
{
   int medio;
   if (i>j) return 0;
   medio = (i+j) / 2;
    if (A[medio] < X)
            return busqueda binaria(A,X,medio+1,j)
  else if (A[medio] > X)
        return busqueda binaria(A,X,i,medio-1)
else
    return medio;
}
```



Programación Recursiva

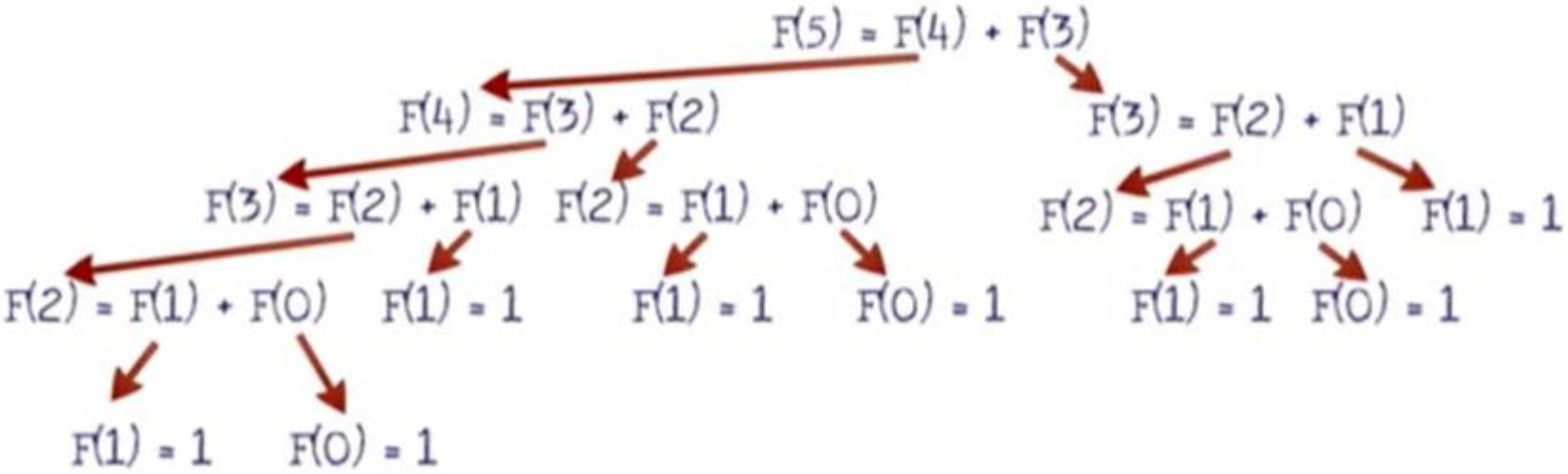
Para el caso de **multiple-división**, sería por ejemplo, recorrer un árbol binario no vacío en **pre-orden**. Para ello, hay que realizar las siguientes operaciones recursivamente en cada nodo, comenzando con el nodo de raíz:

1. Primero se accede a la raíz
2. Si el hijo izquierdo no está vacío, recorrer el subárbol izquierdo.
3. Si el hijo derecho no está vacío, recorrer el subárbol derecho.

```
public void preorden(Nodo a)
  {
     if(!vacioArbol(a))
     {
        System.out.print(a.getDato()+ " ");
        preorden(getIzqArbol(a));
        preorden(getDerArbol(a));
     }
  }
```

Programación Recursiva

Conclusión



Tema 2 Programación Recursiva

Estrategias de Programación y Estructuras de Datos

CA Guadalajara (UNED)