



Computación I

Curso 2023

Facultad de Ingeniería
Universidad de la República

Programas Recursivos

Redundancia

- Función Fibonacci

- $\text{Fib}(1) = 1$

- $\text{Fib}(2) = 2$

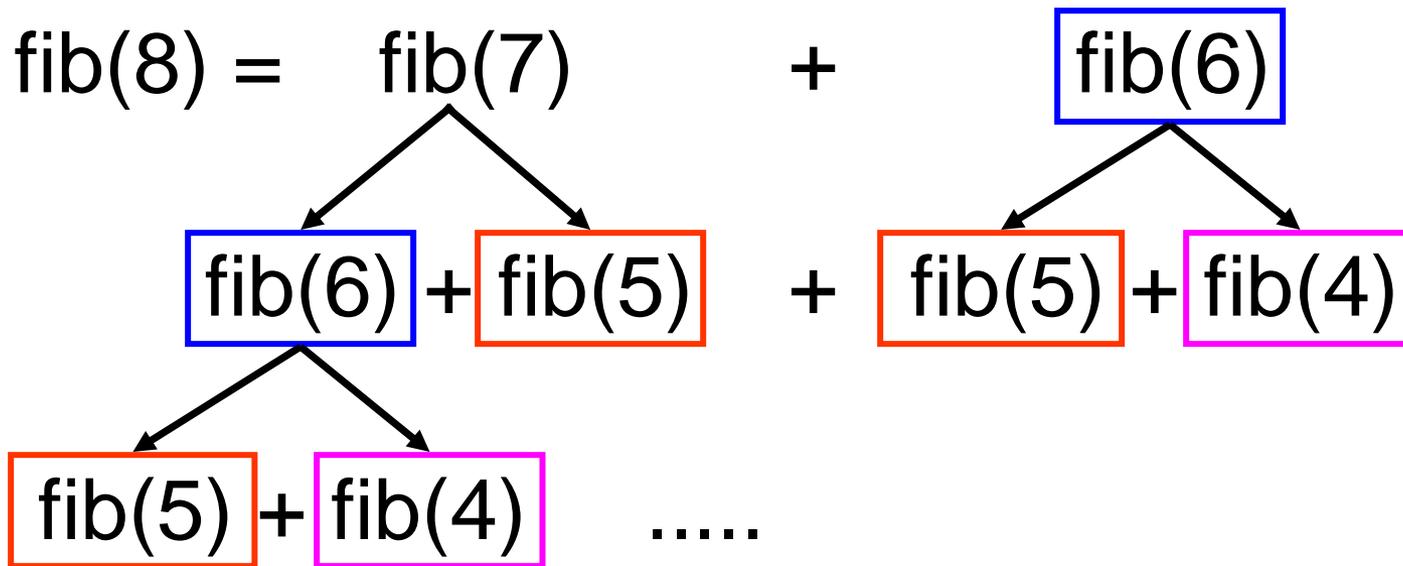
- $\text{Fib}(n) = \text{Fib}(n-1) + \text{Fib}(n-2)$

- Programa Recursivo

```
function fn=fib(n)
if n == 1
    fn = 1;
elseif n == 2
    fn = 2;
else
    fn = fib(n-1) + fib(n-2);
end
```

Programas Recursivos

Redundancia



- Redundancia en cálculos

Recursión vs Iteración

- No siempre es conveniente utilizar una solución recursiva.
 - Problemas de memoria.
 - Se podría cancelar la ejecución por falta de memoria.
 - Redundancia en cálculos.

Recursión vs Iteración

■ Recursiva

```
function fn=fib(n)
if n == 1
    fn = 1;
elseif n == 2
    fn = 2;
else
    fn = fib(n-1) + fib(n-2);
end
```

Recursión vs Iteración

■ Iterativa

```
function fn=fib(n)
if n==1
    fn=1;
elseif n==2
    fn=2;
else
    penult=1;
    ult=2;
    for i=3:n
        nue = penult + ult;
        penult = ult;
        ult = nue;
    end
    fn = ult;
end
```

■ Evita cálculos repetitivos

- Utiliza variables auxiliares para almacenar resultados intermedios

■ Menos clara.

■ Más eficiente.

Recursión vs Iteración

- **Ventajas de la Recursión.**
 - Soluciones simples y claras a problemas inherentemente recursivos.
- **Desventajas de la Recursión: INEFICIENCIA**
 - Memoria
 - Nueva asignación de memoria en cada llamada recursiva.
 - Procesamiento
 - Redundancia en cálculos.
- **Claridad vs. Sobrecarga**

Ejercicio

Escribir un programa recursivo que calcule los cuadrados de los enteros hasta N.

Ejercicio - Solución

```
function y = Cuadrados (n)
if n == 0
    y = [0];
else
    y = [Cuadrados (n-1)  n*n];
end
```

Programas Recursivos

Ejemplo

- Búsqueda recursiva en un array
 - Posibles casos base
 - Vector vacío
 - Vector con un elemento
 - Posible caso recursivo
 - Vector con más de un elemento
 - En cada llamada recursiva reducir el tamaño del vector

Programas Recursivos

Ejemplo

```
function fn=Pertenece(n,a)
```

```
if isempty(a)  
    fn=0;
```

Casos Base

```
else
```

```
    if a(1)==n  
        fn=1;
```

Vector más Pequeño

```
    else
```

```
        fn=Pertenece(n,a(2:length(a)));
```

```
    end
```

```
end
```

Caso Recursivo