



Grado en Ingeniería Información

Estructura de Datos y Algoritmos

Sesión 5

Curso 2023-2024

Marta N. Gómez

6. Funciones recursivas:

Ejemplo:

```
long Fibonacci (int num)
{  if ((num == 1) || (num == 2)) return (1);
    else return (Fibonacci(num-1) + Fibonacci(num-2));
}
```

$$T(n) = 2 + T(n-1) + T(n-2) \leq 2 + 2T(n-1)$$

6. Funciones recursivas:

Ejemplo:

```
long Fibonacci (int num)
{  if ((num == 1) || (num == 2)) return (1);
    else return (Fibonacci(num-1) + Fibonacci(num-2));
}
```

$$T(n) = 2 + T(n-1) + T(n-2) \leq 2 + 2T(n-1) =$$

$$2 + 2(2 + T(n-2) + T(n-3)) \leq 2 + 2(2 + 2T(n-2)) = 2 + 4 + 4T(n-2)$$

6. Funciones recursivas:

Ejemplo:

```
long Fibonacci (int num)
{  if ((num == 1) || (num == 2)) return (1);
    else return (Fibonacci(num-1) + Fibonacci(num-2));
}
```

$$T(n) = 2 + T(n-1) + T(n-2) \leq 2 + 2T(n-1) =$$

$$2 + 2(2 + T(n-2) + T(n-3)) \leq 2 + 2(2 + 2T(n-2)) = 2 + 4 + 4T(n-2) =$$

$$2 + 4 + 4(2 + T(n-3) + T(n-4)) \leq 2 + 4 + 4(2 + 2T(n-3)) =$$

$$2 + 4 + 8 + 8T(n-3)$$

6. Funciones recursivas:

Ejemplo:

```
long Fibonacci (int num)
{  if ((num == 1) || (num == 2)) return (1);
    else return (Fibonacci(num-1) + Fibonacci(num-2));
}
```

$$T(n) = 2 + T(n-1) + T(n-2) \leq 2 + 2T(n-1) =$$

$$2 + 2(2 + T(n-2) + T(n-3)) \leq 2 + 2(2 + 2T(n-2)) = 2 + 4 + 4T(n-2) =$$

$$2 + 4 + 4(2 + T(n-3) + T(n-4)) \leq 2 + 4 + 4(2 + 2T(n-3)) =$$

$$2 + 4 + 8 + 8T(n-3) \leq \dots \leq 2^1 + 2^2 + 2^3 + \dots + 2^{n-1} + 2^{n-1}T(n-(n-1))$$

6. Funciones recursivas:

Ejemplo:

```
long Fibonacci (int num)
{  if ((num == 1) || (num == 2)) return (1);
    else return (Fibonacci(num-1) + Fibonacci(num-2));
}
```

$$\begin{aligned}
 T(n) &= 2 + T(n-1) + T(n-2) \leq 2 + 2T(n-1) = \\
 &2 + 2(2 + T(n-2) + T(n-3)) \leq 2 + 2(2 + 2T(n-2)) = 2 + 4 + 4T(n-2) = \\
 &2 + 4 + 4(2 + T(n-3) + T(n-4)) \leq 2 + 4 + 4(2 + 2T(n-3)) = \\
 &2 + 4 + 8 + 8T(n-3) \leq \dots \leq 2^1 + 2^2 + 2^3 + \dots + 2^{n-1} + 2^{n-1}T(n-(n-1)) = \\
 &\dots = \left(\sum_{i=1}^{n-1} 2^i\right) + 2^{n-1} T(1)
 \end{aligned}$$

6. Funciones recursivas:

Ejemplo:

```
long Fibonacci (int num)
{
    if ((num == 1) || (num == 2)) return (1);
    else return (Fibonacci(num-1) + Fibonacci(num-2));
}
```

$$\begin{aligned}
 T(n) &= 2 + T(n-1) + T(n-2) \leq 2 + 2T(n-1) = \\
 &2 + 2(2 + T(n-2) + T(n-3)) \leq 2 + 2(2 + 2T(n-2)) = 2 + 4 + 4T(n-2) = \\
 &2 + 4 + 4(2 + T(n-3) + T(n-4)) \leq 2 + 4 + 4(2 + 2T(n-3)) = \\
 &2 + 4 + 8 + 8T(n-3) \leq \dots \leq 2^1 + 2^2 + 2^3 + \dots + 2^{n-1} + 2^{n-1}T(n-(n-1)) = \\
 &\dots = \left(\sum_{i=1}^{n-1} 2^i\right) + 2^{n-1} T(1) = \left(\sum_{i=1}^{n-1} 2^i\right) + 2^{n-1} \cdot 2
 \end{aligned}$$

6. Funciones recursivas:

Ejemplo:

```
long Fibonacci (int num)
{
    if ((num == 1) || (num == 2)) return (1);
    else return (Fibonacci(num-1) + Fibonacci(num-2));
}
```

$$\begin{aligned}
 T(n) &= 2 + T(n-1) + T(n-2) \leq 2 + 2T(n-1) = \\
 &2 + 2(2 + T(n-2) + T(n-3)) \leq 2 + 2(2 + 2T(n-2)) = 2 + 4 + 4T(n-2) = \\
 &2 + 4 + 4(2 + T(n-3) + T(n-4)) \leq 2 + 4 + 4(2 + 2T(n-3)) = \\
 &2 + 4 + 8 + 8T(n-3) \leq \dots \leq 2^1 + 2^2 + 2^3 + \dots + 2^{n-1} + 2^{n-1}T(n-(n-1)) = \\
 &\dots = (\sum_{i=1}^{n-1} 2^i) + 2^{n-1} T(1) = (\sum_{i=1}^{n-1} 2^i) + 2^{n-1} 2 = \\
 &(\frac{2^{(n-1)+1} - 2}{2-1}) + 2^n = 2^n - 2 + 2^n = 2^{n+1} - 2
 \end{aligned}$$

6. Funciones recursivas:

Ejemplo:

```
long Fibonacci (int num)
{   if ((num == 1) || (num == 2)) return (1);
    else return (Fibonacci(num-1) + Fibonacci(num-2));
}
```

$$\begin{aligned}
 T(n) &= 2 + T(n-1) + T(n-2) \leq 2 + 2T(n-1) = \\
 &2 + 2(2 + T(n-2) + T(n-3)) \leq 2 + 2(2 + 2T(n-2)) = 2 + 4 + 4T(n-2) = \\
 &2 + 4 + 4(2 + T(n-3) + T(n-4)) \leq 2 + 4 + 4(2 + 2T(n-3)) = \\
 &2 + 4 + 8 + 8T(n-3) \leq \dots \leq 2^1 + 2^2 + 2^3 + \dots + 2^{n-1} + 2^{n-1}T(n-(n-1)) = \\
 &\dots = \left(\sum_{i=1}^{n-1} 2^i\right) + 2^{n-1} T(1) = \left(\sum_{i=1}^{n-1} 2^i\right) + 2^{n-1} 2 = \\
 &\left(\frac{2^{(n-1)+1} - 2}{2-1}\right) + 2^n
 \end{aligned}$$

6. Funciones recursivas:

Ejemplo:

```
long Fibonacci (int num)
{  if ((num == 1) || (num == 2)) return (1);
    else return (Fibonacci(num-1) + Fibonacci(num-2));
}
```

$$\begin{aligned}
 T(n) &= 2 + T(n-1) + T(n-2) \leq 2 + 2T(n-1) = \\
 &2 + 2(2 + T(n-2) + T(n-3)) \leq 2 + 2(2 + 2T(n-2)) = 2 + 4 + 4T(n-2) = \\
 &2 + 4 + 4(2 + T(n-3) + T(n-4)) \leq 2 + 4 + 4(2 + 2T(n-3)) = \\
 &2 + 4 + 8 + 8T(n-3) \leq \dots \leq 2^1 + 2^2 + 2^3 + \dots + 2^{n-1} + 2^{n-1}T(n-(n-1)) = \\
 &\dots = \left(\sum_{i=1}^{n-1} 2^i\right) + 2^{n-1} T(1) = \left(\sum_{i=1}^{n-1} 2^i\right) + 2^{n-1} 2 = \\
 &\left(\frac{2^{(n-1)+1} - 2}{2-1}\right) + 2^n = 2^n - 2 + 2^n = 2^{n+1} - 2
 \end{aligned}$$

Por tanto, $T(n)$ es $\Theta(2^n)$, de orden exponencial.

6. Funciones recursivas:

Coste de la eficiencia

Ejemplo:

```
long FibonacciI (int num)
{  long f1{1}, f2{1}, fn{f1};
  for (int i{3}; i<=num; i++){
    fn = f1 + f2;
    f1 = f2;
    f2 = fn;
  }
  return fn;
}
```

← 3 pasos

← $2(n-2) + 2$ pasos

← 2 pasos (n-2) veces

← 1 paso (n-2) veces

← 1 paso (n-2) veces

← 1 paso

6. Funciones recursivas:

Coste de la eficiencia

Ejemplo:

```
long FibonacciI (int num)
{ long f1{1}, f2{1}, fn{f1};
  for (int i{3}; i<=num; i++){
    fn = f1 + f2;
    f1 = f2;
    f2 = fn;
  }
  return fn;
}
```

← 3 pasos

← $2(n-2) + 2$ pasos

← 2 pasos (n-2) veces

← 1 paso (n-2) veces

← 1 paso (n-2) veces

← 1 paso

$$T(n) = 3 + 2 + 2(n-2) + 4(n-2) + 1 = 6 + 6(n-2) = 6n - 12$$

6. Funciones recursivas:

Coste de la eficiencia

Ejemplo:

```
long FibonacciI (int num)
{ long f1{1}, f2{1}, fn{f1};
  for (int i{3}; i<=num; i++){
    fn = f1 + f2;
    f1 = f2;
    f2 = fn;
  }
  return fn;
}
```

Complexity analysis of the code:

- The initialization of `f1`, `f2`, and `fn` is $\Theta(1)$.
- The loop body (calculating `fn`, updating `f1`, and updating `f2`) is $\Theta(1)$ and is executed $(n-2)$ times.
- The loop is $\Theta(n)$.
- The return statement is $\Theta(1)$.
- The overall complexity is $\Theta(n)$.

$$T(n) = 3 + 2 + 2(n-2) + 4(n-2) + 1 = 6 + 6(n-2) = 6n - 12$$

6. Funciones recursivas:

Ejemplo:

```
long FibonacciI (int num)
{ long f1{1}, f2{1}, fn{f1};
  for (int i{3}; i<=num; i++){
    fn = f1 + f2;
    f1 = f2;
    f2 = fn;
  }
  return fn;
}
```

$$T(n) = 3 + 2 + 2(n-2) + 4(n-2) + 1 = 6 + 6(n-2) = 6n - 12$$

Por tanto, $T(n)$ es $\Theta(n)$, de orden lineal.