UCA Universidad de Cádiz

Grado en Ingeniería Informática
Metodología de la Programación

esther.silva@uca.es

# ANÁLISIS DE ALGORITMOS

## Ejercicios propuestos

---

**Objetivos**

- Estudiar la complejidad de algoritmos.
- Analizar el coste temporal de algoritmos.
- Calcular el coste de algoritmos en el mejor y peor caso.

Calcule el orden de complejidad de las siguientes funciones en el mejor y peor caso. Recuerde seleccionar cuidadosamente la operación crítica, obtener exhaustivamente la función $t(n)$ y determinar el orden, explicando detalladamente todo el proceso.

**Ejercicio 1**

```
//Cabecera: entero fact(E entero: n)
//Precondición: n = N ≥ 0
//Postcondición: devuelve N!
entero: función fact(E entero: n)
var
    entero: mult
inicio
    mult ← 1
    mientras n > 0 hacer
        mult ← mult · n
        n ← n − 1
    fin_mientras
    devolver mult
fin_función
```

**Ejercicio 2**

```
//Cabecera: entero multVector(E Vect: v, E entero: n)
//Precondición: v = A[1...n] ∧ n = N > 0
//Postcondición: devuelve ∏_{α=1}^{N} v[α]
entero: función multVector(E entero: n)
var
    entero: mult, i
inicio
    mult ← 1
    desde i ← 1 hasta n hacer
        mult ← mult · v[i]
    fin_desde
    devolver mult
fin_función
```

**Ejercicio 3**

```
//Cabecera: entero squareR(E entero: n)
//Precondición: n = N > 0
//Postcondición: devuelve √n
entero: función squareR(E entero: n)
var
    entero: x
inicio
    x ← 0
    mientras (x + 1) · (x + 1) ≤ n hacer
        x ← x + 1
    fin_mientras
    devolver x
fin_función
```

**Ejercicio 4**

```
//Cabecera: entero maxVector(E Vect: v, E entero: n)
//Precondición: v = A[1...n] ∧ n = N > 0
//Postcondición: el elemento máximo max del vector v: ∀α : 1 ≤ α ≤ n, v[α] ≤ max
entero: función maxVector(E Vect: v, E entero: n)
var
    entero: i, max
inicio
    max ← v[1]
    desde i ← 2 hasta n hacer
        si v[i] > max entonces
            max ← v[i]
        fin_si
    fin_desde
    devolver max
fin_función
```

**Ejercicio 5**

```
entero: función mult(E entero: n)
var entero: res, i, j
inicio
    res ← 1
    i ← 1
    mientras i ≤ n hacer
        j ← n
        mientras j ≥ 1 hacer
            res ← res · j
            j ← j − 3
        fin_mientras
        i ← i + 2
    fin_mientras
    devolver res
fin_función
```

**Ejercicio 6**

```
entero: función sum(E entero: n)
var entero: sum, i, j
inicio
    sum ← 0
    desde i ← 1 hasta n hacer
        j ← 1
        mientras j ≤ n hacer
            sum ← sum + j
            j ← j · 4
        fin_mientras
    fin_desde
    devolver sum
fin_función
```

**Ejercicio 7**

```
entero: función prod(E entero: n)
var entero: res, i, j
inicio
    res ← 1
    i ← n
    mientras i ≥ 1 hacer
        j ← n
        mientras j ≥ 1 hacer
            res ← res · 2
            j ← ⌊j/4⌋
        fin_mientras
        i ← i − 3
    fin_mientras
    devolver res
fin_función
```

**Ejercicio 8**

//Cabecera: prodMatrices(E Mat: x, E Mat: y, E/S Mat: z, E entero: $n$)
//Precondición: $n = N \geq 0$
//Postcondición: devuelve en $z$ el producto de las matrices $x$ e $y$
**procedimiento** prodMatrices(E Mat: x, E Mat: y, E/S Mat: z, E entero: $n$)
**var** entero: $sum, i, j, k$
**inicio**
    **desde** $i \leftarrow 1$ **hasta** $n$ **hacer**
        **desde** $j \leftarrow 1$ **hasta** $n$ **hacer**
            $sum \leftarrow 1$
            **desde** $k \leftarrow 1$ **hasta** $n$ **hacer**
                $sum \leftarrow sum + x[i][k] \cdot y[i][k]$
            **fin_desde**
            $z[i][j] \leftarrow sum$
        **fin_desde**
    **fin_desde**
**end_procedure**

**Ejercicio 9**

//Cabecera: entero sum(E entero: $n$)
//Precondición: $n = N \geq 0$
//Postcondición: devuelve la suma $\sum_{\alpha=1}^{N} \sum_{\beta=0}^{\alpha} \beta$
entero: **función** sum(E entero: $n$)
**var** entero: $res, d, x$
**inicio**
    $res \leftarrow 0$
    $d \leftarrow 1$
    **mientras** $d \leq n$ **hacer**
        $x \leftarrow d$
        **mientras** $x \geq 0$ **hacer**
            $res \leftarrow res + x$
            $x \leftarrow x - 1$
        **fin_mientras**
        $d \leftarrow d + 1$
    **fin_mientras**
    **devolver** $res$
**fin_función**

**Ejercicio 10**

//Cabecera: boolean sym(E Mat: $m$, E entero: $n$)
//Precondición: $n = N \geq 0$
//Postcondición: devuelve verdadero si la matriz $m$ es una matriz simétrica y flaso en caso contrario

```
lógico: función sym(E Mat: m, E entero: n)
var entero: i, j
    lógico: c
inicio
    i ← 1
    c ← true
    mientras i ≤ n ∧ c hacer
        j ← i + 1
        mientras j ≤ n ∧ c hacer
            c ← m[i][j] == m[j][i]
            j ← j + 1
        fin_mientras
        i ← i + 1
    fin_mientras
    devolver c
fin_función
```

**Ejercicio 11**

```
//Cabecera: entero sum(E entero: n)
//Precondición: n = N ≥ 0
//Postcondición: devuelve la suma sum_{α=1}^{√N} sum_{β=0}^{α} β
entero: función sum(E entero: n)
var entero: res, d, x
inicio
    res ← 0
    d ← 1
    mientras d · d ≤ n hacer
        x ← d
        mientras x ≥ 0 hacer
            res ← res + x
            x ← x − 1
        fin_mientras
        d ← d + 1
    fin_mientras
    devolver res
fin_función
```

**Ejercicio 12**

```
entero: función sum(E entero: n)
var entero: res, i, j
inicio
    res ← 0
    desde i ← 1 hasta n hacer
        j ← i
        mientras j ≤ n hacer
            j ← j^2
            res ← res + j
        fin_mientras
    fin_desde
```

```
    devolver res
fin_función
```

**Ejercicio 13**

```
entero: función sum(E entero: n)
var entero: res, i, j, k
inicio
    res ← 0
    desde i ← 1 hasta n hacer
        desde j ← i + 1 hasta n hacer
            desde k ← i + j hasta n hacer
                res ← res + 1
            fin_desde
        fin_desde
    fin_desde
    devolver res
fin_función
```

**Ejercicio 14**

```
entero: función sum(E entero: n)
var entero: res, i, j, k
inicio
    res ← 0
    desde i ← 1 hasta n hacer
        desde j ← i · i hasta n hacer
            desde k ← 1 hasta ⌊n/2⌋ hacer
                res ← res + 1
            fin_desde
        fin_desde
    fin_desde
    devolver res
fin_función
```

**NOTA.-** Se supone la existencia de los tipos Vect y Mat definidos como:

vector[$N$] de entero: Vect
matriz[$N$,$N$] of entero: Mat

siendo $n \leq N$.