Departamento de Ingeniería Informática
Grado en Ingeniería Informática
Elisa Guerrero Vázquez
Esther L. Silva Ramírez

Metodología de la Programación
Anexo Tema 4 – Teoría
ANÁLISIS DE ALGORITMOS
Algoritmo de búsqueda y ordenación

Búsqueda secuencial en un vector

tipo
    vector [n] de entero: Vect

// Cabecera: busq_secuencial(E Vect: v, E entero: n, E entero: elem)
// Precondición: $n > 0 \wedge n \in \mathbb{N}$. v es un vector de n elementos y elem $\in \mathbb{N}$.
// Postcondición: Devuelve la posición de la primera aparición de elem en el vector v ó n+1 si dicho elemento no se encuentra en el vector.

```
entero función busq_secuencial (E Vect: v, E entero: n, E
                                entero: elem)
var
    entero : i, elemento
inicio
    i ←1
    mientras (v[i] ≠ elem) ∧ (i ≤ n) hacer
        i ← i + 1
    fin_mientras
    devolver i
    fin_función
```

Búsqueda secuencial en un vector ordenado

tipo
vector [n] de entero: Vect

// Cabecera: busq_secuencial_ord(E Vect: v, E entero: n, E entero: elem)
// Precondición: $n > 0 \wedge n \in \mathbb{N}$. v es un vector de n elementos ordenados ascendentemente y elem $\in \mathbb{N}$.
// Postcondición: Devuelve la posición de la primera aparición de elem en el vector v ó n+1 si dicho elemento no se encuentra en el vector.

```
entero función busq_secuencial _ord(E Vect:v, E entero: n, E
                                    entero: elem)
var
    entero : i, elemento
inicio
    i ←1
    mientras (v[i] < elem) ∧ (i < n) hacer
        i ← i + 1
    fin_mientras
   si (v[i]=elem) entonces
        devolver i
   si_no
       devolver n+1
   fin_si
    fin_función
```

Búsqueda Dicotómica (El vector debe estar ordenado)

```
tipo
    vector [n] de entero: Vect

// Cabecera: busq_binaria_iter (E Vect: v, E entero: n, E entero:
   elem, E entero: izqda, E entero: drcha)
 // Precondición: n > 0 ∧ n ∈ ℕ . v es un vector de n  elementos y
    elem ∈ ℕ .
 // Postcondición: Devuelve la posición de la primera aparición
    de elem en el vector v ó n+1 si dicho elemento no se encuentra
    en el vector.
entero función busq_binaria_iter(E Vect: v, E entero: n, E
    entero: elem)
var
    entero: central, pos, izqda, drcha
```

inicio

izqda ← 1

drcha ← n

central ← $\left\lfloor \frac{\text{izqda} + \text{drcha}}{2} \right\rfloor$

mientras v[central] ≠ elem ∧ izqda < drcha **hacer**

si elem > v[central] **entonces**

izqda ← central+1

si_no

drcha ← central -1

fin_si

central ← $\left\lfloor \frac{\text{izqda} + \text{drcha}}{2} \right\rfloor$

fin mientras

si v[central]=elem **entonces** pos ← central

si_no pos ← n+1

devolver pos

fin_función

//Cabecera: busqueda_binaria(E/S Vect: v, E entero: n , E entero: elem)
//Precondición: $n > 0 \wedge n \in \mathbb{N}$. v es un vector de n elementos y elem $\in \mathbb{N}$.
//Postcondición: Devuelve la posición de la primera aparición de elem en el vector v ó n+1 si dicho elemento no se encuentra en el vector.

entero **función** busqueda_binaria(E/S Vect: v, E entero: n , E entero: elem)
inicio
devolver busq_binaria_iter (v, n, elem, 1, n)
fin_función

Búsqueda Dicotómica (El vector debe estar ordenado)

```
tipo
      vector [n] de entero: Vect

// Cabecera: busq_binaria_rec (E Vect: v, E entero: n, E entero:
   n, E entero: elem, E entero: izqda, E entero: drcha)
 // Precondición: n > 0 ∧ n ∈ℕ . v es un vector de n elementos y
   elem ∈ℕ .
 // Postcondición: Devuelve la posición de la primera aparición
   de elem en el vector v ó n+1 si dicho elemento no se encuentra
   en el vector.
entero función busq_binaria_rec(E Vect: v, E entero: n, E
      entero: elem, E entero: izqda, E entero: drcha)
var
   entero: central, pos
```

inicio

si (izqda > drcha) **entonces** pos ← n+1

si_no

central ← $\left\lfloor \frac{\text{izqda} + \text{drcha}}{2} \right\rfloor$

según_sea elem **hacer**

elem> v[central]:

devolver busq_binaria_rec (v, n, elem, central+1, drcha)

elem < v[central]:

devolver busq_binaria_rec (v, n, elem, izqda, central-1)

elem=v[central]: pos ← central

fin_según

devolver pos

fin_si

fin_función

//Cabecera: busqueda_binaria(E/S Vect: v, E entero: n , E entero: elem)
//Precondición: $n > 0 \wedge n \in \mathbb{N}$. v es un vector de n elementos y elem $\in \mathbb{N}$.
//Postcondición: Devuelve la posición de la primera aparición de elem en el vector v ó n+1 si dicho elemento no se encuentra en el vector.

entero **función** busqueda_binaria(E/S Vect: v, E entero: n , E entero: elem)
inicio
  devolver busq_binaria_rec(v, n, elem, 1, n)
fin_función

Método de la Burbuja o Intercambio Directo

tipo

vector [n] de entero: Vect

// Cabecera: burbuja (E/S Vect: v, E entero: n)
// Precondición: $n > 0 \wedge n \in \mathbb{N}$. v es un vector de n elementos.
// Postcondición: Vector v con los elementos ordenados ascendentemente.

procedimiento burbuja (E/S Vect: v, E entero: n)

var

entero: i, j, aux

inicio

```
desde i ←1 hasta n-1 hacer
   desde j ←1 hasta n-i hacer
      si v[j] > v[j+1] entonces
         aux ← v[j+1]
         v[j+1] ← v[j]
         v[j] ← aux
      fin_si
   fin_desde
fin_desde
fin_procedimiento
```

Método de la Burbuja Mejorado

```
tipo
      vector [n] de entero: Vect

// Cabecera: burbuja_mejorado (E/S Vect: v, E entero: n)
// Precondición: n > 0 ∧ n ∈ ℕ . v es un vector de n elementos.
// Postcondición: Vector v con los elementos ordenados
   ascendentemente.

procedimiento burbuja_mejorado(E/S Vect: v, E entero: n)
var
      entero: i, j, aux
      lógico: no_interc
inicio
```

```
        i ←1
        repetir
            no_interc ← verdadero
            desde j ←1 hasta n-i hacer
                si v[j ] > v[j+1] entonces
                    aux ← v[j+1]
                    v[j+1] ← v[j]
                    v[j] ← aux
                    no_interc ← falso
                fin_si
            fin_desde
            i ←i+1
        hasta nointerc
fin_procedimiento
```

Método de Ordenación rápida

// Cabecera: ordenación_rápida_rec(E entero: izqda, E entero: dcha, E/S Vect: v)

// Precondición: izqda, drcha $> 0 \wedge$ izqda, drcha $\in \mathbb{N}$. v es un vector de n elementos.

// Postcondición: Vector v con los elementos ordenados ascendentemente.

procedimiento ordenación_rápida_rec(E entero: izqda, E entero: drcha, E/S Vect: v)

var

entero: i, j, central, pivote,aux

inicio

i ← izqda

j ← drcha

central ← $\left\lfloor \frac{i+j}{2} \right\rfloor$ //Tomando como pivote el elemento central

pivote ← v[central]

mientras (i ≤ j) **hacer**

mientras (v[j] > pivote) **hacer**

j← j - 1

fin_mientras

mientras (v[i] < pivote) **hacer**

i ← i + 1

fin_mientras

```
            si (i ≤ j) entonces
                aux ← v[j]
                v[j] ← v[i]
                v[i] ← aux
                i ← i + 1
                j ← j -1
            fin_si
        fin_mientras
        si (izqda < j) entonces
            ordenación_rápida_rec(izqda , j, v)
        fin_si
        si (drcha > i) entonces
            ordenación_rápida_rec(i, drcha, v)
        fin_si
fin_procedimiento
```

```
//Cabecera: ordenación_rápida(E/S Vect: v, E entero: n)
//Precondición: n > 0 ∧ n ∈ ℕ . v es un vector de n elementos.
//Postcondición: Vector v con los elementos ordenados
   ascendentemente.

procedimiento ordenación_rápida (E/S Vect: v, E entero: n)
inicio
   ordenación_rápida_rec(1, n, v)
fin_procedimiento
```

Método de Ordenación por inserción

```
// Cabecera: inserción(E/S Vect: v, E entero: n)
// Precondición: n > 0 ∧ n ∈ℕ . v es un vector de n elementos.
// Postcondición: Vector v con los elementos ordenados
    ascendentemente.

procedimiento inserción (E/S Vect: v, E entero: n)
var
    entero: i, j, temp
inicio
```

```
    desde i ← 2 hasta n hacer
        j ← i-1
        temp ← v[i]
        mientras j > 0 ∧ temp < v[j]
            v[j+1] ← v[j]
            j ← j-1
        fin_mientras
        v[j+1] ← temp
    fin_desde
fin_procedimiento
```

Método de Ordenación por selección

```
// Cabecera: seleción(E/S Vect: v, E entero: n)
// Precondición: n > 0 ∧ n ∈ℕ . v es un vector de n elementos.
// Postcondición: Vector v con los elementos ordenados
    ascendentemente.

procedimiento selección (E/S Vect: v, E entero: n)
var
    entero: i, j, i_min, temp
inicio
```

```
desde i ← 1 hasta n-1 hacer
        i_min ← i
        desde j ← i+1 hasta n hacer
            si  v[j] < v[i_min]
                i_min ← j
            fin_si
        fin_desde
        si i ≠ i_min
            temp ← v[i]
            v[i] ← v[i_min]
            v[i_min] ← temp
        fin_si
    fin_desde
fin_procedimiento
```