UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escola Tècnica Superior d'Enginyeria Informàtica

etsinf

Tema 1. Recursividad

Programación (PRG)

Jorge González Mollá

Departamento de Sistemas Informáticos y Computación

Índice

Búsqueda ascendente

[…] int busquedaAscendente(tipoBase[] a, int inicio, int fin)

- Llamada inicial: busquedaAscendente(a, 0, a.length-1), donde inicio se instancia a 0 y fin a a.length-1.
- Asumimos un Caso Base de Longitud 0:

```
/** 0 <= inicio <= a.length              Caso Base de Longitud 0
  * fin == a.length-1 */
[…] int busquedaAscendente(tipoBase[] a, int inicio, int fin) {
    if (inicio>fin) { // Condición de Caso Base de Longitud 0
       // array sin elementos:          Caso Base de Longitud 0
       return -1;  // Fracaso en la búsqueda
    }
    else {
       if (propiedad(a[inicio])) return inicio;
          // comprueba si a[inicio] cumple la propiedad enunciada
       else return busquedaAscendente(a, inicio+1, fin);
    }
}
```

Búsqueda descendente

[…] int busquedaDescendente(tipoBase[] a, int inicio, int fin)

- Llamada inicial: busquedaDescendente(a, 0, a.length-1), donde inicio se instancia a 0 y fin a a.length-1.
- Asumimos un Caso Base de Longitud 0:

```
/** inicio == 0
  * -1 <= fin <= a.length-1              Caso Base de Longitud 0 */
[…] int busquedaDescendente(tipoBase[] a, int inicio, int fin) {
    if (inicio>fin) { // Condición de Caso Base de Longitud 0
       // array sin elementos:          Caso Base de Longitud 0
       return -1;  // Fracaso en la búsqueda
    }
    else {
       if (propiedad(a[fin])) return fin;
          // comprueba si a[fin] cumple la propiedad enunciada
       else return busquedaDescendente(a, inicio, fin-1);
    }
    }
}
```

Método lanzadera

- Se suele definir un método público homónimo, llamado guía o lanzadera, el cual hace la llamada inicial al método recursivo sobre todo el array `a`. Esta técnica permite ocultar la estructura recursiva asociada a los parámetros `inicio`/`fin` del perfil del método recursivo correspondiente, el cual ahora lógicamente se puede (y se debería) definir como `private`.

```
public […] int busquedaAscendente(tipoBase[] a) {
        // return busquedaAscendente(a, 0, a.length-1)
}

public […] busquedaDescendente(tipoBase[] a) {
        // return busquedaDescendente(a, 0, a.length-1)
}
```

Búsqueda ascendente simplificada

- Sustituir las referencias al parámetro `fin` por su valor inicial `a.length-1`:

```
[…] int busquedaAscendenteSimplificada(tipoBase[] a, int inicio)
```

- Llamada inicial: `inicio` se instancia a 0

  `busquedaAscendenteSimplificada(a, 0)`:

- Asumimos un Caso Base de Longitud 0:

```
/** 0 <= inicio <= a.length          Caso Base de Longitud 0 */
[…] int busquedaAscendenteSimplificada(tipoBase[] a, int inicio) {
    if (inicio >= a.length) { // Condición CB de Longitud 0
       // array sin elementos:        C. Base de Longitud 0
       return -1;  // Fracaso en la búsqueda
    }
    else {
       if (propiedad(a[inicio])) return inicio;
          // comprueba si a[inicio] cumple la propiedad enunciada
       else return busquedaAscendenteSimplificada(a, inicio+1);
    }
}
```

Búsqueda descendente simplificada

- Sustituir las referencias al parámetro `inicio` por su valor inicial `0`:

```
[…] int busquedaDescendenteSimplificada(tipoBase[] a, int fin)
```

- Llamada inicial: `fin` se instancia a `a.length-1`

```
busquedaDescendenteSimplificada(a, a.length-1)
```

- Asumimos un Caso Base de Longitud 0:

```
/** -1 <= fin <= a.length-1          Caso Base de Longitud 0 */
[…]int busquedaDescendenteSimplificada(tipoBase[] a, int fin) {
    if (fin < 0) {      // Condición Caso Base de Longitud 0
       // array sin elementos:          C. Base de Longitud 0
       return -1;  // Fracaso en la búsqueda
    }
    else {
       if (propiedad(a[fin])) return fin;
          // comprueba si a[fin] cumple la propiedad enunciada
       else return busquedaDescendenteSimplificada(a, fin-1);

    }
}
```

Ejemplo: Buscar el primero (ascendente)

1. Obtener la posición del primer entero distinto de 0 en a[0..a.length-1]

```
/** 0 <= inicio <= a.length */
public static int encontrarA(int[] a, int inicio)
```

Búsqueda ascendente simplificada

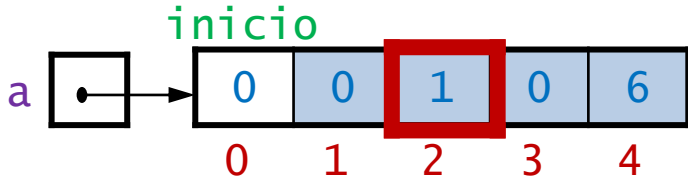
2. Caso Base de Longitud 0 (inicio >= a.length): fracasa y devuelve -1.

   Caso General: En otro caso, el resultado será:

   si a[inicio] es distinto de 0 devuelve inicio

   en otro caso, la búsqueda continúa sobre a[inicio+1..a.length-1]

Llamada inicial: encontrarA(a, 0)




```
/** 0 <= inicio <= a.length */
public static int encontrarA(int[] a, int inicio) {
        if (inicio >= a.length) return -1;
        else if (a[inicio] !=0) return inicio;
        else return encontrarA(a, inicio+1);
}
```

3. En el caso general, en cada llamada el parámetro inicio se incrementa en 1; eventualmente llegará a valer a.length (alcanzando por tanto el caso base), o bien se verificará la propiedad de búsqueda, finalizando así el algoritmo. Para cualquiera de las llamadas, el índice inicio cumple la precondición del método.

Ejemplo: Buscar el último (descendente)

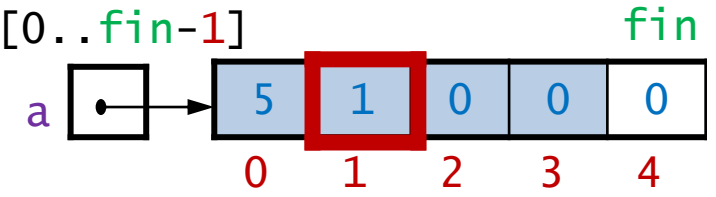
1. Obtener la posición del último entero distinto de 0 en a[0..a.length-1]

Búsqueda recursiva descendente

```
/** -1 <= fin <= a.length-1 */
public static int encontrarD(int[] a, int fin)
```

2. Caso Base de Longitud 0 (fin < 0): fracasa y devuelve -1
   Caso General: En otro caso, el resultado será:
   si a[fin] es distinto de 0 devuelve fin
   en otro caso, la búsqueda continúa sobre a[0..fin-1]

Llamada inicial: encontrarD(a, a.length-1)




```
/** -1 <= fin <= a.length-1 */
public static int encontrarD(int[] a, int fin) {
        if (fin==-1) return -1;
        else if (a[fin] !=0) return fin;
        else return encontrarD(a, fin-1);
}
```

3. En el caso general, en cada llamada el parámetro fin disminuye de 1 en 1; eventualmente llegará a valer −1 (alcanzando por tanto el caso base), o bien se verificará la propiedad de búsqueda finalizando de este modo el algoritmo. En cualquiera de las llamadas, el índice fin satisface la precondición del método.