Universidad
Carlos III de Madrid

Tema 9

Algoritmos sobre listas

Programación

2015-2016

Tema 9. Algoritmos sobre listas

- **Algoritmos sobre Arrays.**
  - **Búsqueda.**
  - **Inserción.**
  - **Ordenación.**

Algoritmos sobre Arrays

- En la vida coditiana es frecuente la búsqueda de elementos en conjuntos o listas de datos. Por ejemplo: Índices de libros, callejeros, guías de teléfonos, etc.
- Es evidente que si dichos conjuntos de datos no estuvieran ordenados la búsqueda sería larga y pesada.
- En computación se dedica bastante tiempo a la búsqueda y ordenación de datos. Por ejemplo accesos a Google, etc.
- En este tema vamos a abordar algunos de los algoritmos de búsqueda y ordenación más utilizados sobre arrays.

Tema 9. Algoritmos sobre listas

- **Algoritmos sobre Arrays.**
  - **Búsqueda.**
  - **Inserción.**
  - **Ordenación.**

Búsqueda

Los algoritmos de búsqueda más sencillos son:

- la búsqueda **secuencial**
- la búsqueda **binaria**

Búsqueda secuencial

- Puede realizarse en arrays ordenados y en arrays desordenados.
- En los desordenados es la única posible.
- Hay que comparar cada elemento del array con el elemento buscado.
- La búsqueda finaliza cuando se encuentra el elemento o cuando se llega al final del array.
- Si se ha encontrado el elemento se indicará su posición en el array, y en caso contrario se dará un mensaje de error.
- Es un método sencillo, pero poco eficiente. Obsérvese que puede ser necesario recorrer el array entero.

Algoritmo

Variables
lista[ ] es un array de enteros
elemento es un entero el dato buscado
Pseudocódigo
Inicio programa
mientras (elemento no encontrado y pos<=ultimo) hacer
        si elemento == lista [pos] entonces
                encontrado ← cierto
            si no
                pos ← pos+1
        fin si
fin-mientras
si encontrado == cierto entonces
        devolver pos
    si no
        Visualizar “Elemento no encontrado”
Fin programa

Código Java

```
public static int secuencial (int [] vector,int elem, int p,int u){
        // recibe una matriz no ordenada, un elemento a buscar
        // y los limites del rango donde buscar (p y u)
        // devuelve la posición del elemento o -1 si no esta
        int i;
        boolean encontrado;
        i =p;
        encontrado = false;
        while (i<=u && !encontrado){
           if (elem == vector[i]){
              encontrado = true;
           }
           i++;
        }
        if (encontrado){
           return i;
        } else {
          return -1;
        }
    }
```

Búsqueda binaria

El array debe estar ordenado.

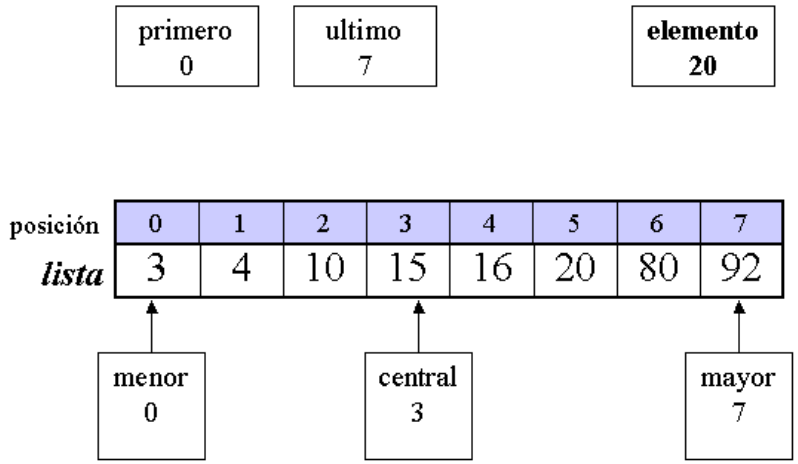
Es mas rápido que el secuencial.

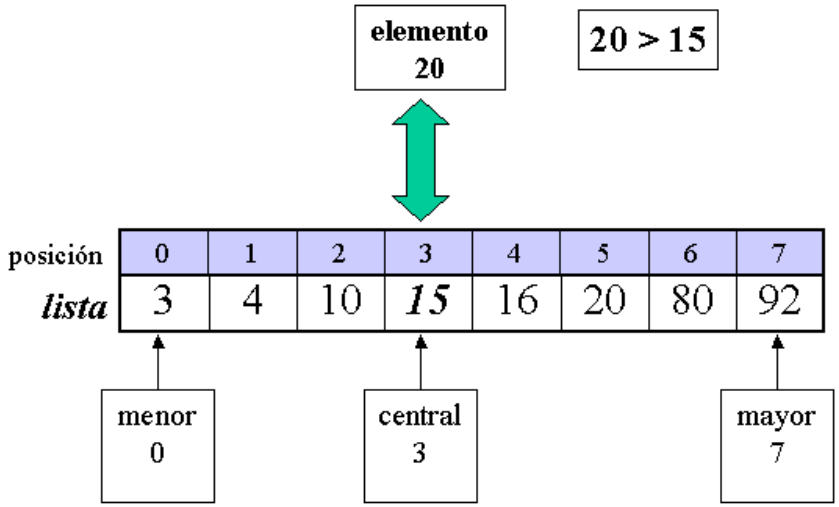
Algoritmo:

- Se busca el elemento central del array, si éste no coincide con el elemento buscado se determina en que mitad del array debe estar. Se busca en esa mitad repitiendo el proceso las veces que haga falta hasta encontrar el elemento, o bien hasta que se determine que no está en la lista.

Búsqueda binaria

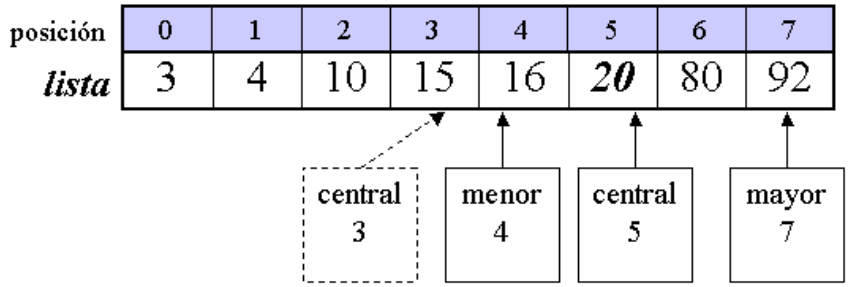
Buscamos el elemento 20 en el array *lista*

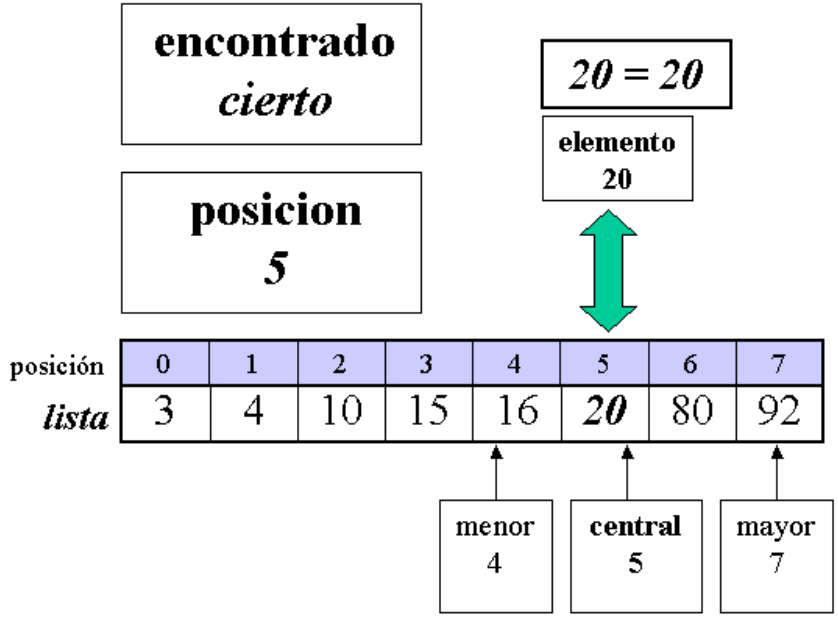




Búsqueda binaria

Buscamos el elemento 20 en el array *lista*





Algoritmo

Variables
lista(i) es un array ordenado
men, may, cen son variables enteras para el índice en la lista
X es el valor búscado
Pseudocódigo
Inicio programa.
men ← 1
may ← n
cen ← (n+1)/2
Mientras lista(cen) <> X y men < may hacer
    si lista(cen) > X entonces
        may ← cen-1
    sino
        men ← cen+1
    fin_si
    cen ← (men+may)/2
Fin mientras
Si lista(cen) = X entonces
    escribir “Encontrado en la posición”, cen
sino
    escribir “No encontrado en el array”
Fin si
Fin Programa

Código Java

```
public static void binaria (int [] vector,int elem, int p,int u, int posicion, boolean encontrado){
    // recibe una matriz ordenada, un elemento a buscar y los limites del rango donde buscar(p y u)
     // recibe dos referencias con un booleano para devolver si el elemento se encuentra en la matriz,
     // y con un entero para devolver la posicion donde esta o la que le corresponde si no esta
    int i;
     int menor = p, mayor = u, medio=0;
     encontrado = false;
     while (!encontrado && mayor >= menor){
                        medio = (mayor + menor) / 2;
                        if (elem == vector[medio])
                                encontrado=true;
                        else if (elem > vector[medio])
                 // buscar en la mitad superior
                                menor = medio + 1;
                        else
                       // buscar en la mitad inferior
                                mayor = medio - 1;
            }
            if (encontrado)
                        posicion = medio;
            else
                        posicion = menor;
    }
```

Tema 9. Algoritmos sobre listas

- **Algoritmos sobre Arrays.**
  - **Búsqueda.**
  - **Inserción.**
  - **Ordenación.**

Inserción

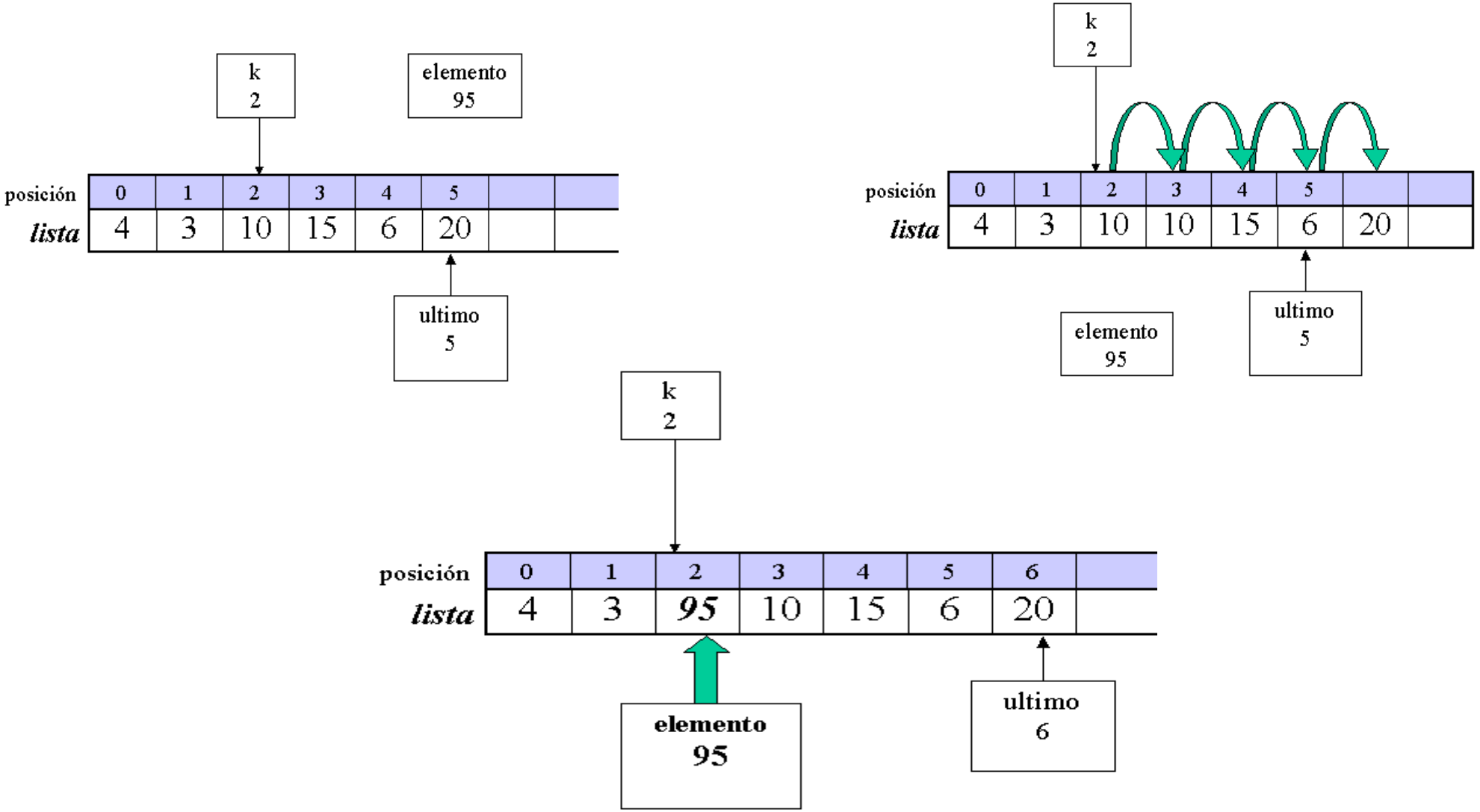
El array debe tener espacio para añadir un nuevo elemento.

Dos posibilidades:

- Array desordenado
  - El elemento se insertara en el lugar que se indique.
- Array ordenado
  - El elemento ocupara el lugar que le corresponda.

Inserción en array desordenado.

Insertar el elemento 95 en la posición 2 del array *lista*



Inserción en array desordenado. Algoritmo

Variables

j índice del array; k posición de inserción

elemento valor a insertar

lista(j) array de elementos

Pseudocódigo

```
Inicio Programa
si lista(j)  llena entonces
    escribir “Error, no se puede insertar “
fin_si
para j = ultimo hasta j>=k ; con incremento -1  hacer
    lista(j+1) ← lista(j)
fin_para
lista(k) ← elemento
ultimo ← ultimo + 1
Fin programa
```

Código Java

```
public static int insertarNoOrdenada (int [] vector, int elemento,
                                      int pos, int ultimo){
// recibe una matriz, un elemento a insertar, y la posicion de
// insercion. También recibe un objeto num de tipo Entero con la última
// posición ocupada de la matriz. Se supone que el valor de
// pos es correcto
    if (ultimo == vector.length - 1)
       return -1; // codigo de error, matriz llena
    else {
       for (int i = ultimo ; i >= pos; i-- )
           vector[i+1] = vector[i];
       vector[pos] = elemento;
       ultimo ++;
       return 0;
    }
}  // fin de insertarNoOrdenada
```

Inserción en array ordenado

- Si se va a insertar en un array ordenado hay que insertar el elemento en el lugar que le corresponde.

- Se utilizará el algoritmo de búsqueda binaria para buscar la posición de inserción.

- No obstante caben dos posibilidades:
  - Que no se admitan elementos repetidos
  - Que pueda haber elementos repetidos.

Inserción en array ordenado sin repetidos

- Se comprueba que el array no está lleno
- Se utiliza la búsqueda binaria para comprobar si el elemento a insertar ya está en el array.
- Si está se lanza un mensaje de error y se termina.
- Si el elemento no está, el mismo algoritmo de búsqueda binaria nos devuelve la posición de inserción, y se procede como en caso de array desordenado.

Inserción en array ordenado con repetidos

- Se comprueba que el array no está lleno.
- Se utiliza la búsqueda binaria para encontrar la posición en la que hay que insertar el elemento, y se procede como en caso de array desordenado.

Tema 9. Algoritmos sobre listas

- **Algoritmos sobre Arrays.**
  - **Búsqueda.**
  - **Inserción.**
  - **Ordenación.**

Métodos de ordenación

- Intercambio directo
- Inserción directa
- Selección directa

Ordenación por intercambio directo
Método de la burbuja

Se intercambian pares de elementos consecutivos.

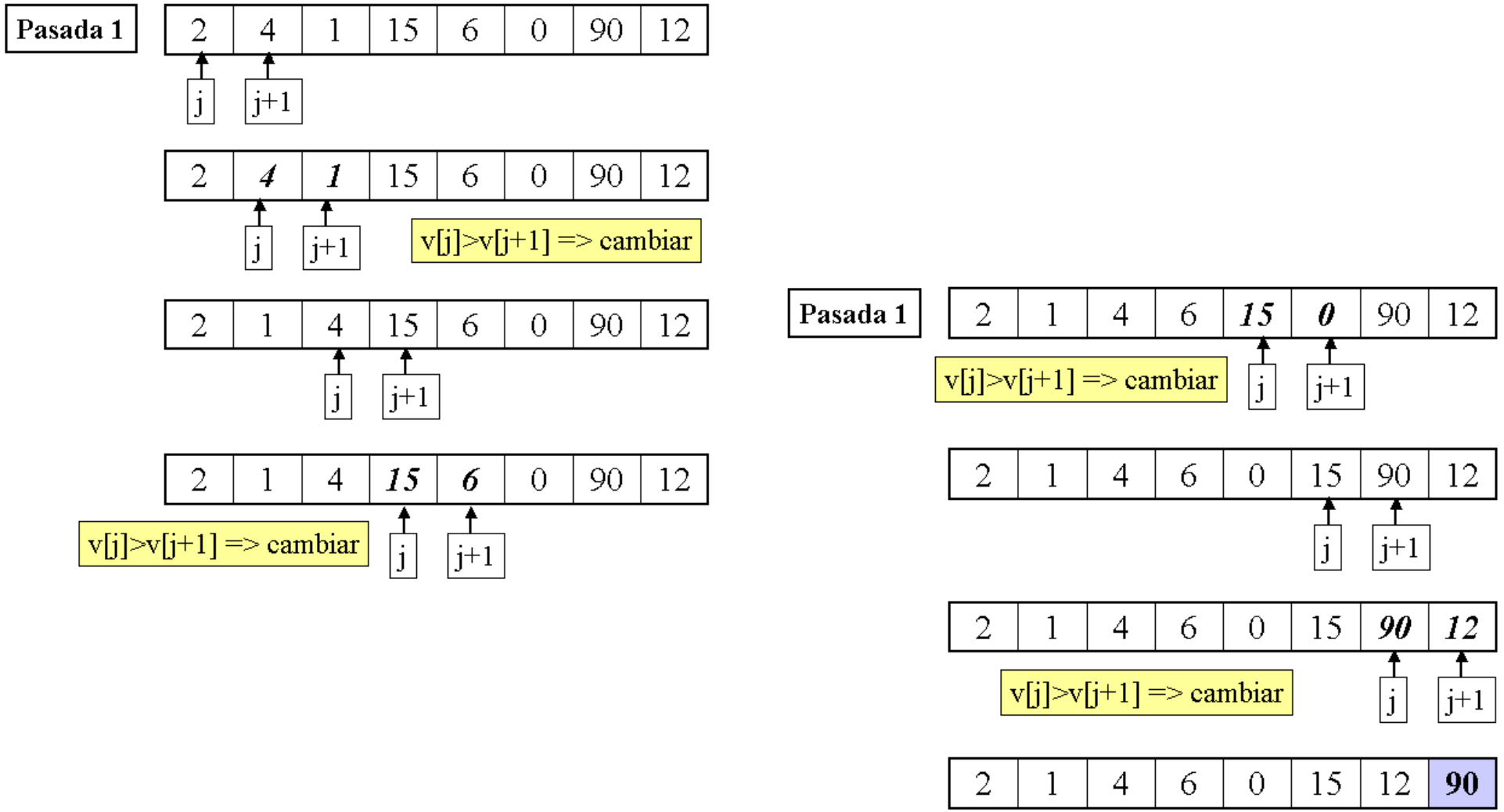
Consiste en recorrer varias veces el array comparando en cada una de ellas los elementos consecutivos para intercambiarlos si no están ordenados.

Existen varias versiones:

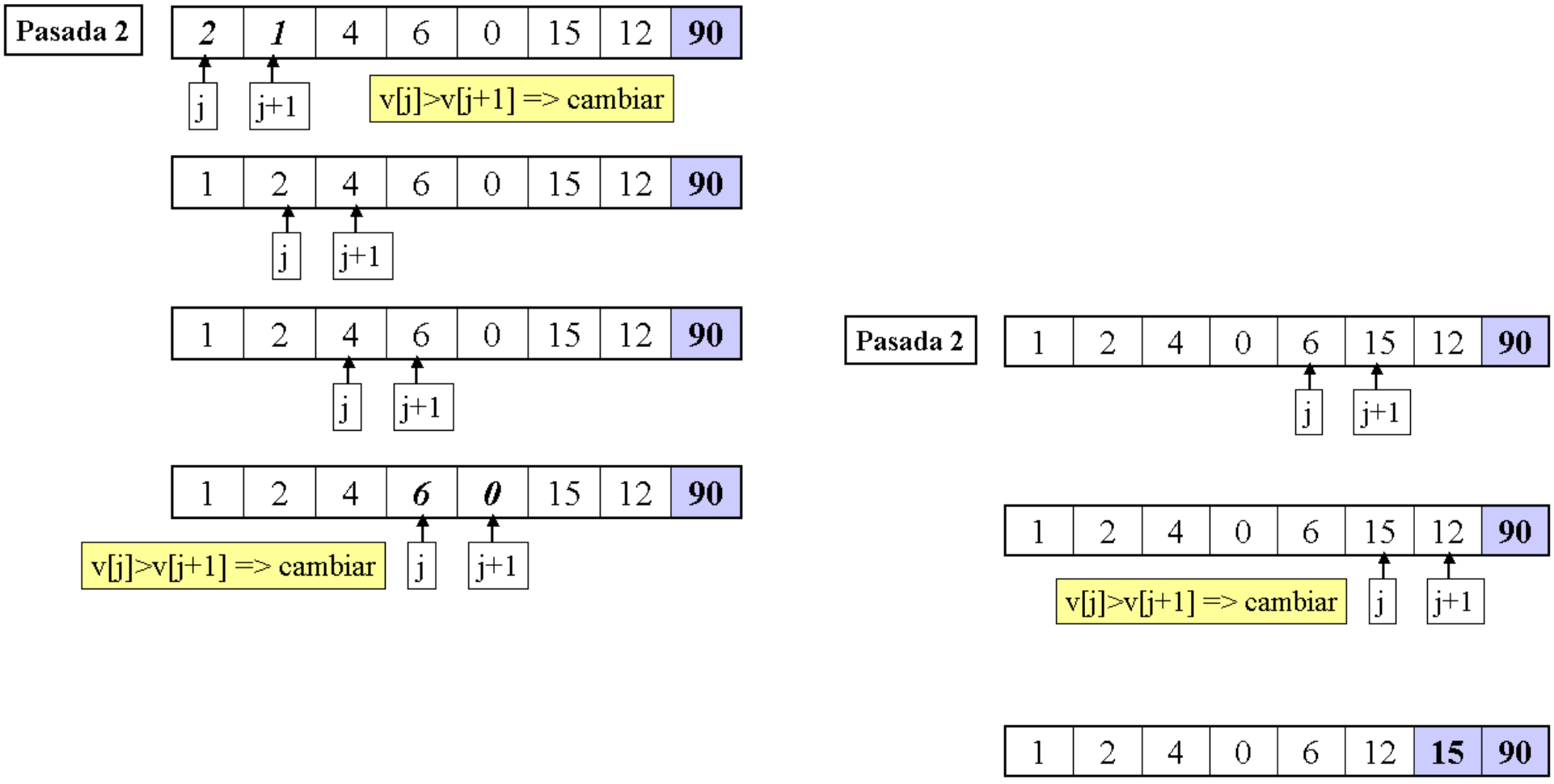
- Recorrer el vector de izquierda a derecha
- Recorrer el vector de derecha a izquierda.

En los dos casos el orden es ascendente.

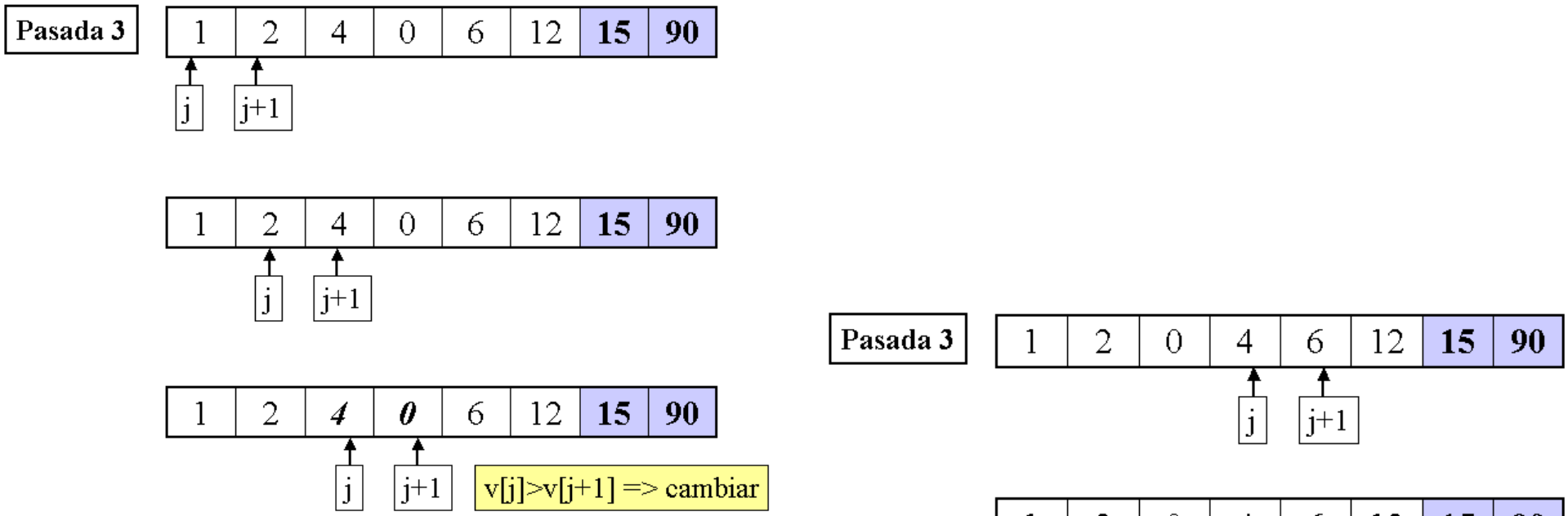
Método de la burbuja



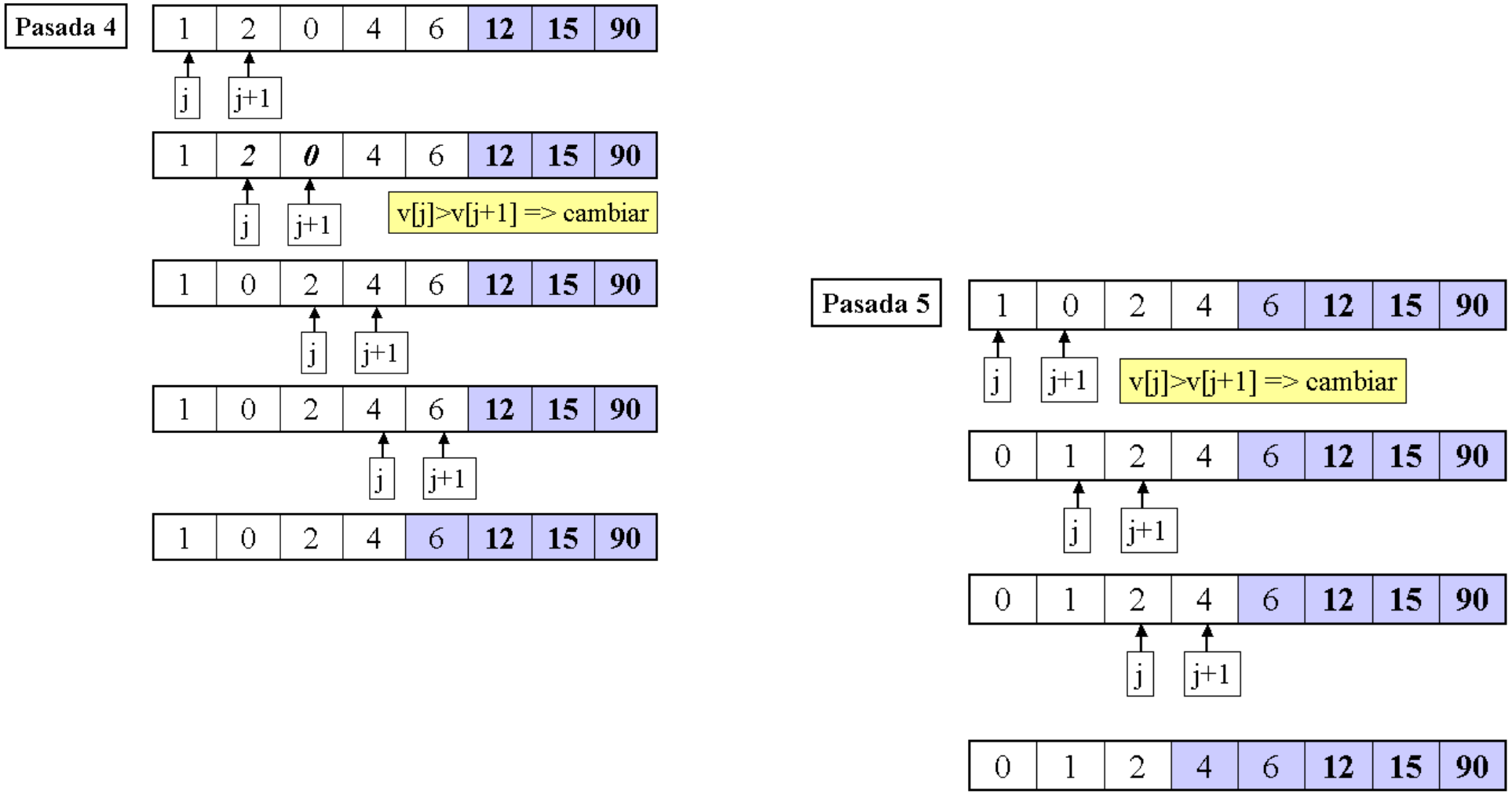
Método de la burbuja



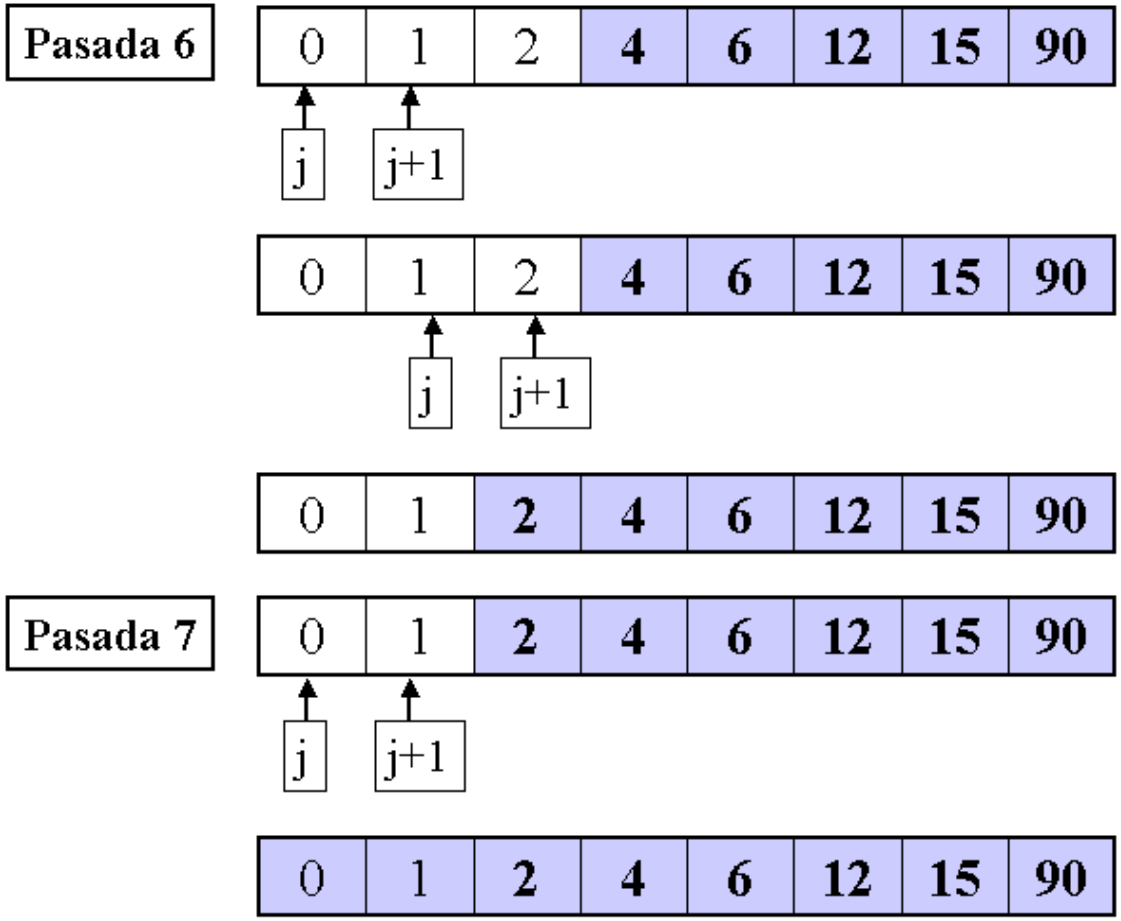
Método de la burbuja



Método de la burbuja



Método de la burbuja



Método de la burbuja. Algoritmo

Variables

P, I índices de bucles
n tamaño del array
Lista array de elementos a ordenar
Aux variable del tipo de elementos del array

Pseudocódigo

```
Inicio Programa
Para P de 1 a  n-1 hacer
    Para I de 1  a   n-P  hacer
    Si Lista(I) > Lista(I+1) entonces
    Aux ← Lista(I)
    Lista(I) ← Lista(I+1)
    Lista(I+1) ← Aux
    Fin_si
    Fin_para
Fin_para
Fin Programa
```

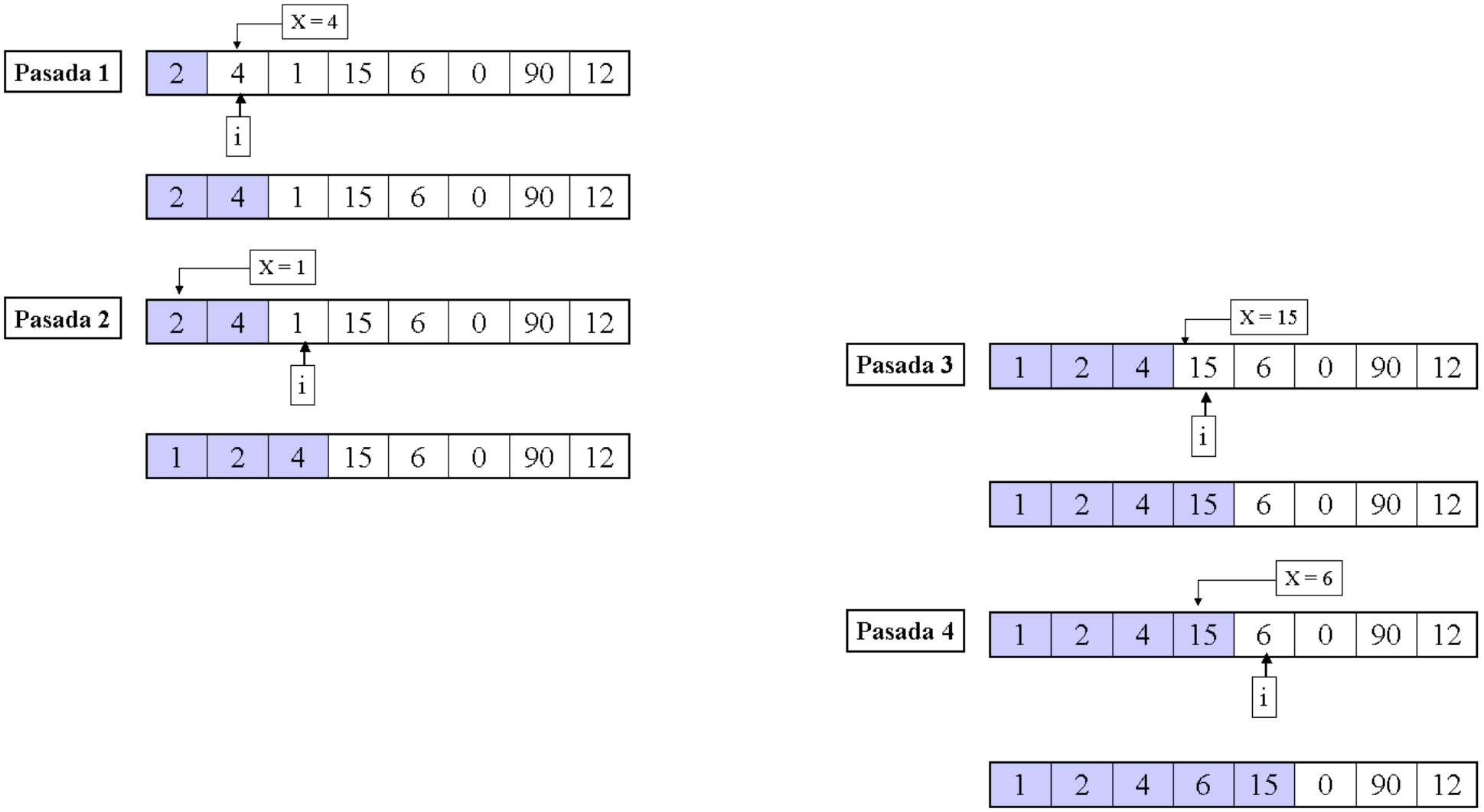
Método de la burbuja. Código

```
public static void burbuja(int [] vector, int num){

   // ordena por el algoritmo de la burbuja una matriz de
   // num elementos utiles.

   int aux;
   for (int k=1; k<num; k++)
     for (int j=0; j<num - k; j++)
       if (vector[j] > vector [j+1]){
         aux = vector [j];
         vector [j] = vector [j+1];
         vector [j+1] = aux;
       }
}
```

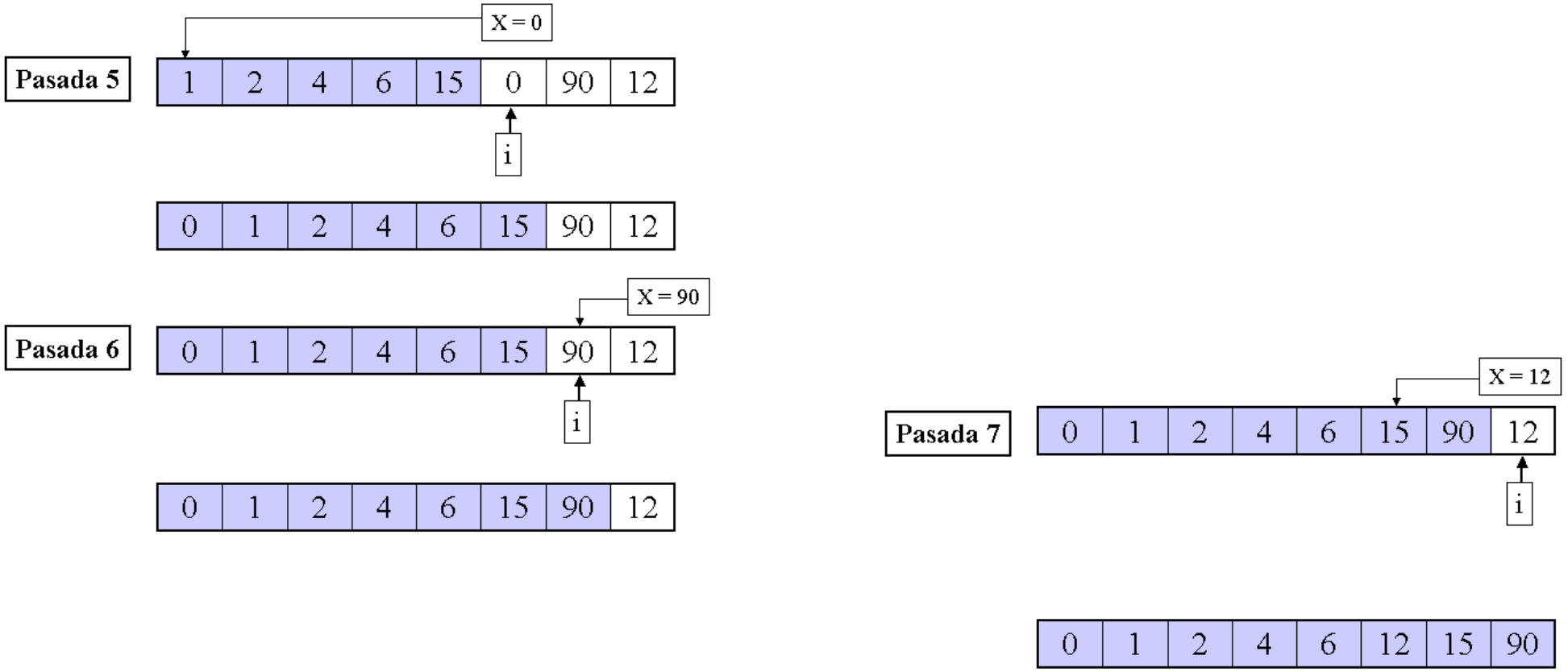
Ordenación por inserción directa
Método de la baraja

Se trata de insertar cada elemento, empezando en el segundo y hasta el último, en el lugar que le corresponde en el grupo ordenado que se va creando a su izquierda.

Método de la baraja



Método de la baraja



Método de la baraja. Algoritmo

Variables

I, J índices de bucles
n tamaño del array
Lista array de elementos a ordenar
AUX variable del tipo de elementos del array

Pseudocódigo

```
Inicio Programa
Para I de 2 a n  hacer
    AUX ← Lista(I)
    J ← I - 1
    Mientras Lista(J) > AUX  y J > 1 hacer
        Lista(J+1) ← Lista(J)
        J ← J - 1
    Fin_mientras
    Si Lista(J) > AUX  entonces
       Lista(J+1) ← Lista(J)
       Lista(J) ← AUX
    sino
       Lista(J) ← AUX
    Fin_si
Fin_para
Fin Programa
```

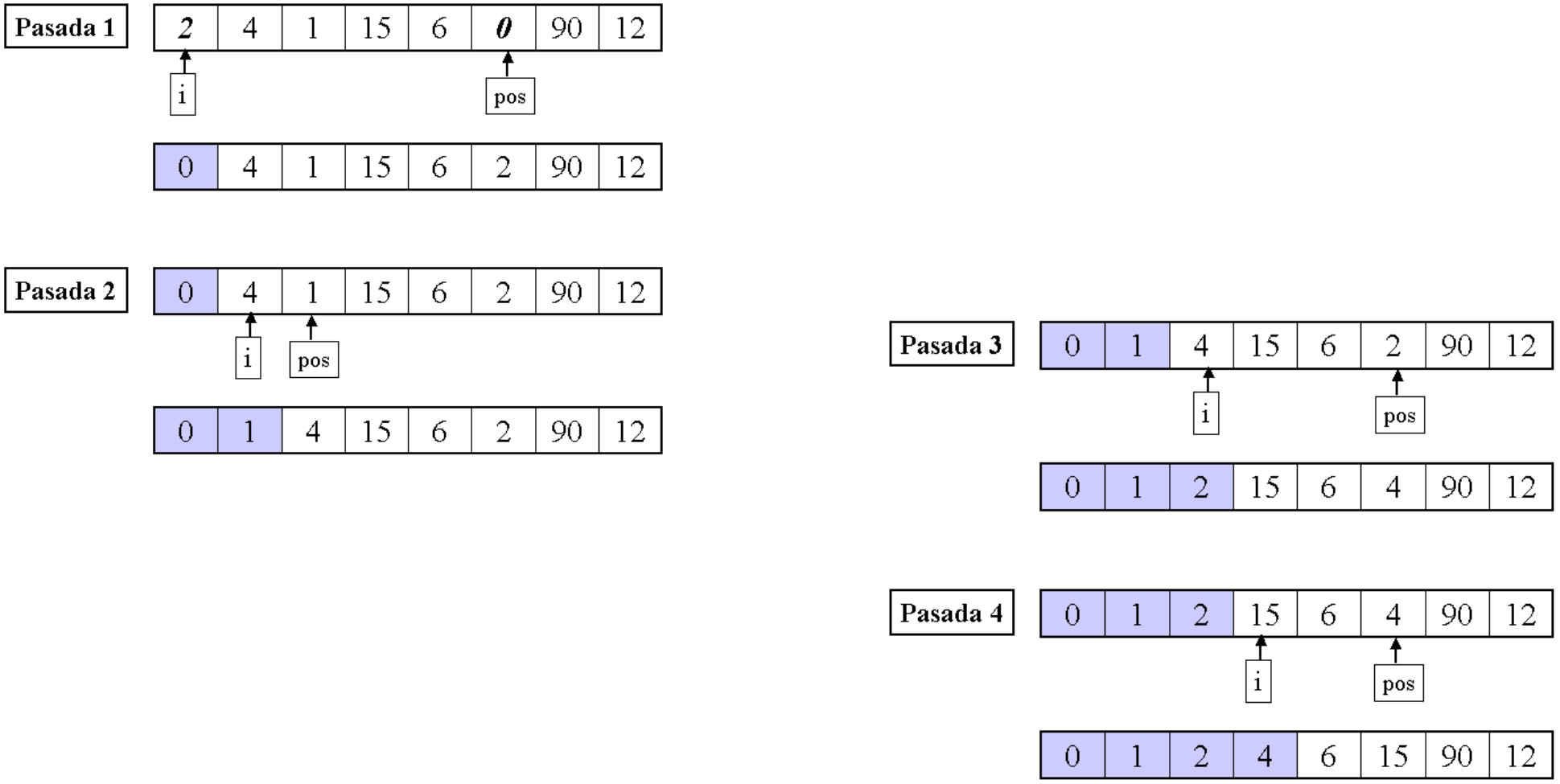
Método de la baraja. Código

```
public static void insercionDirecta(int [] lista, int num){
  // ordena por el algoritmo de inserción directa una matriz de
  // num elementos útiles.
    int i, j;
    int aux;
    for (i = 1; i<num; i++){
      aux = lista[i];
      j= i - 1;
      while (j >= 0 &&  aux < lista[j]) {
          lista [j + 1] = lista [j];
           j--;
      }
      lista [j + 1] = aux;
    }
  }
```

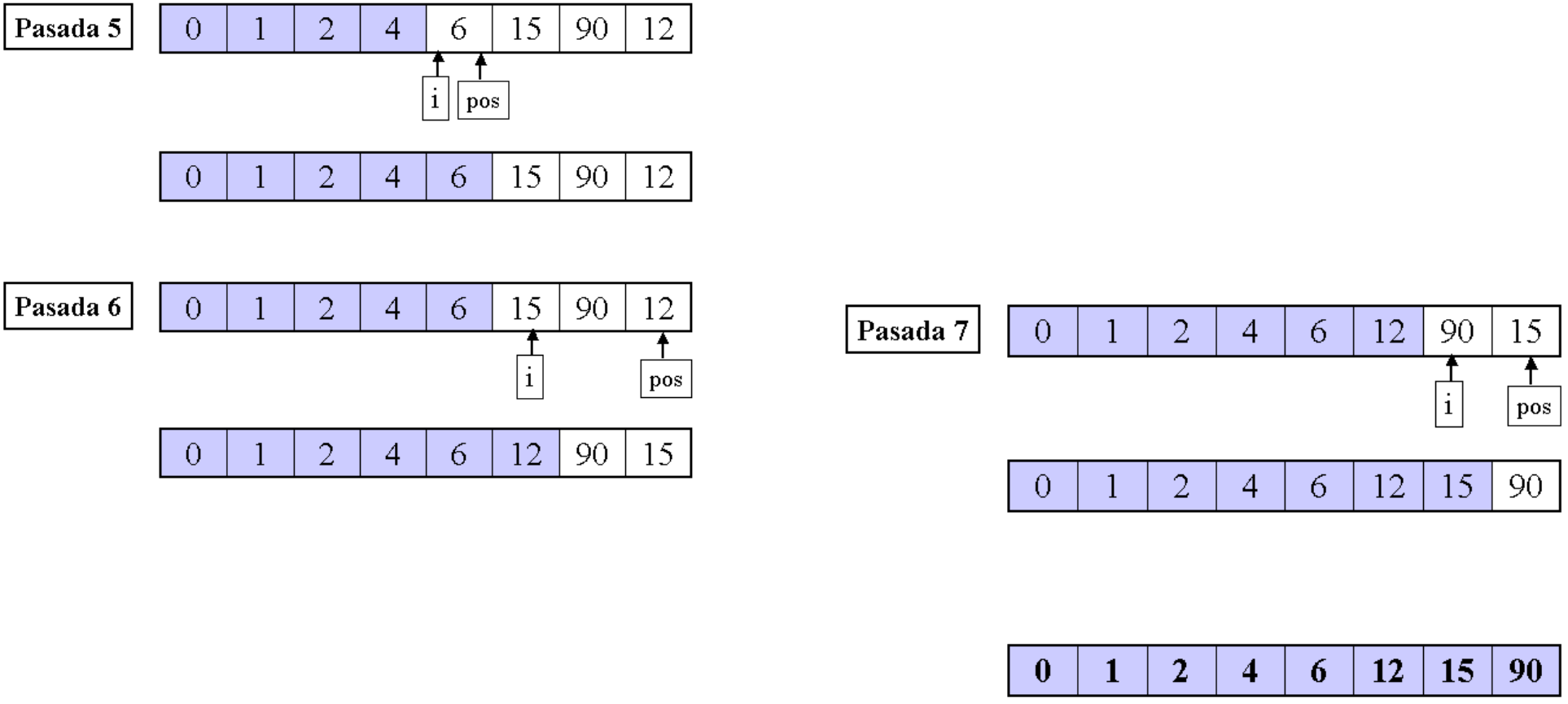
Método de selección directa

- Se selecciona el elemento que contiene el valor menor del array y se intercambia con el primer elemento del array.
- A continuación se repite la operación con los elementos desde el segundo al último, y el menor se intercambia con el del segundo elemento.
- Tenemos dos sublistas: la de la izquierda que va quedando ordenada y la de la derecha es la que hay que ordenar.
- Así se continua hasta que sólo queda un elemento que ya estará ordenado por ser el último.

Ordenación por selección directa



Ordenación por selección directa



Ordenación por selección directa. Algoritmo

Variables

Lista Array de elementos para ordenar de tamaño n
I, J, K Índices de los bucles
AUX variable del mismo tipo que el array

Pseudocódigo

```
Inicio Programa
    Para I de 1 a (n-1) hacer
        K ← I
        AUX ← Lista(I)
        Para J de I+1 a n  hacer
            Si Lista(J) < AUX entonces
              K ← J
              AUX ← Lista(J)
            Fin_si
        Fin_para
        Lista(K) ← Lista(I)
        Lista(I) ← AUX
    Fin_para
Fin Programa
```

Ordenación por selección directa. Código

```
public static void seleccionDirecta(int [] vector, int n){
  // ordena por el algoritmo de inserción directa una matriz de
  // n elementos útiles.
        int menor;
        int i, j, pos;
        for (i = 0; i < n-1; i++){
            menor = vector[i];
            pos = i;
            // buscamos el menor de la sublista derecha
            for (j = i+1; j < n ; j++)
                if (vector[j] < menor){
                    menor = vector[j];
                    pos = j;
                }
            vector[pos] = vector[i];
            vector[i] = menor;
        }
  }
```