Estructura de datos y Algoritmos

Tema II

Clasificación en memoria principal

2.1. Introducción.

2.2. Métodos de clasificación directos:

2.2.1. Clasificación por inserción:

- Directa
- Binaria.

2.2.2. Clasificación por selección directa.

2.2.3. Clasificación por intercambio:

- Directo (método de la burbuja)
- Método por sacudida

2.3. Métodos de clasificación avanzados:

2.3.1. Inserción por incremento decreciente (Shell).

2.3.2. Clasificación por montón (Floyd)

2.3.3. Clasificación por partición (clasif. rápida)

- Obtención del k-ésimo menor elemento.

2.1. Introducción.

- Tipos de Clasificación:
  - Clasificación interna (Estructuras en Mem. Principal)
  - Clasificación externa (Estructuras en Mem. Secundaria)

Interna	Externa
En Memoria Primaria: rápida	En Memoria Secundaria: lenta
Dispone todos los datos simult.	Dispone unos pocos a la vez

Cosas a tener en cuenta en la elección del algoritmo

- 1. Tamaño del conjunto de datos.
- 2. Eficacia. Depende de (1).
- 3. Estabilidad:
  - El orden relativo de los elementos con iguales claves permanece inalterado en el proceso de clasificación, es decir, no intercambia elementos con claves iguales.
- 4. Comportamiento natural:
  - Si el mejor caso es que el arreglo esté inicialmente ordenado.
- 5. Tipo de clasificación:
  - interna o externa.

Medida de la eficacia:

- Depende de:
  - Número C de comparaciones.
  - Número M de movimientos.
- Si ORDEN:
  - n^2 Métodos Directos
- Si ORDEN:
  - n * log (n) Métodos Avanzados
  - (Siendo "n" el número de elementos a clasificar)
- Analizamos
  - El mejor caso
  - El promedio
  - El peor caso

2.2. Métodos de clasificación directos

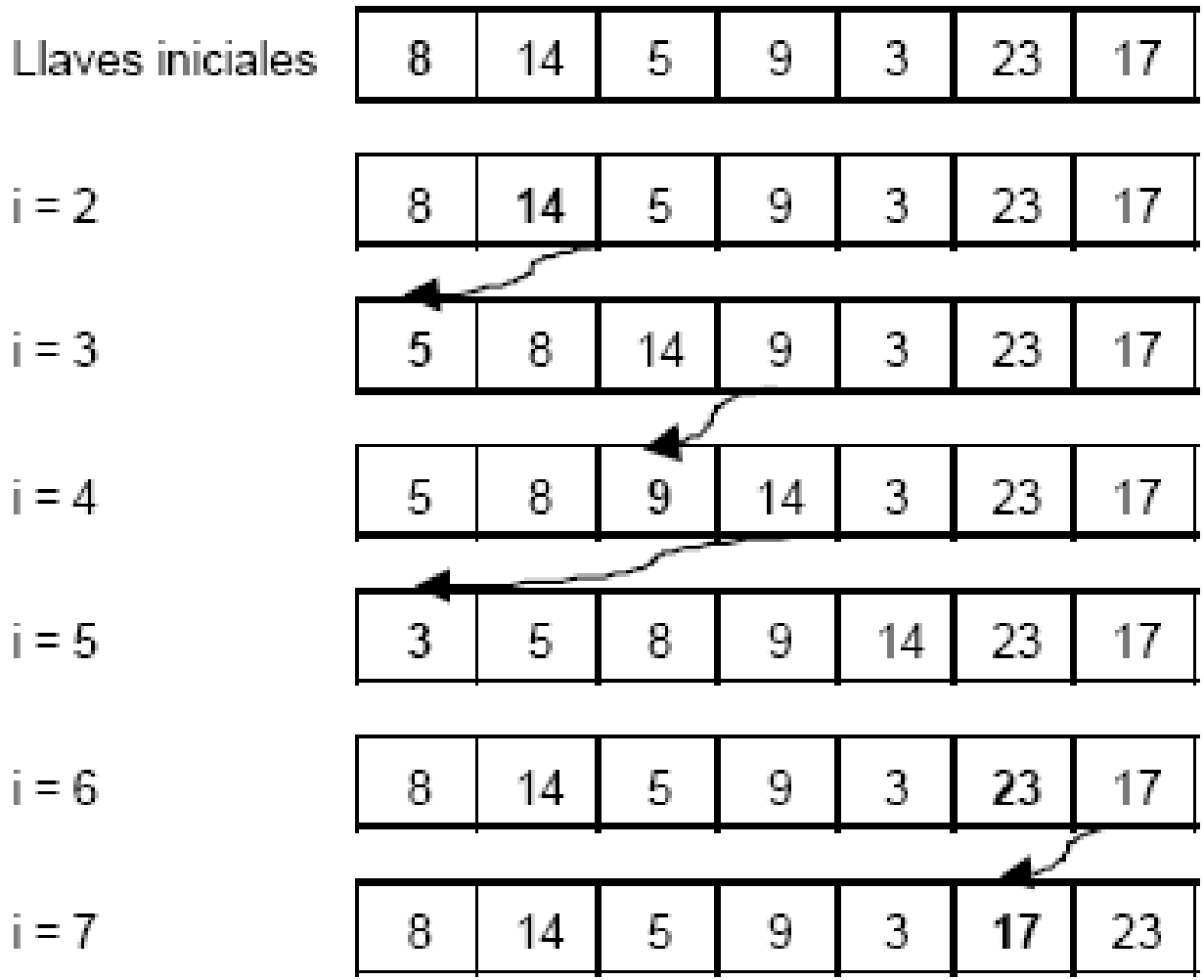
- Las colecciones de elementos a clasificar se almacenarán en arreglos
- Clasificación por inserción:
  - Consiste en insertar un elemento dado en el lugar que le corresponda
    - Directa
    - Binaria
- Clasificación por selección directa.
- Clasificación por intercambio:
  - Directo (método de la burbuja)
  - Método por sacudida

Clasificación por inserción directa

- En un primer paso se considera que la parte ordenada esta formada por el primer elemento del arreglo. El procedimiento será:
  - Tomar un elemento en la posición i (i= 2,3,…n)
  - Buscar su lugar en las posiciones anteriores (parte ordenada)
  - Mover hacia la derecha los restantes
  - Insertarlo.

Clasificación por inserción directa.

```
BEGIN
    FOR i := 2 TO n DO
        a[0]:= a[i]; (*garantiza la salida del bucle while*)
        j := i;
        WHILE a[0]< a[j-1] DO
            a[j] := a[j-1];
            j := j-1
        END;
        a[j] := a[0] (*inserción*)
    END
END
```


Llaves iniciales
8 14 5 9 3 23 17
i = 2
8 14 5 9 3 23 17
i = 3
5 8 14 9 3 23 17
i = 4
5 8 9 14 3 23 17
i = 5
3 5 8 9 14 23 17
i = 6
8 14 5 9 3 23 17
i = 7
8 14 5 9 3 17 23

Clasificación por inserción binaria

- Mejora el método de inserción.
- Para ello se toma el elemento que ocupa la posición central de la parte ordenada y se compara con el elemento a insertar.. Si es mayor se descarta la parte izquierda y si es menor la derecha. Aplicando sucesivamente el mismo proceso se obtiene la posición donde se debe insertar:
  - Tomar un elemento de la posición i
  - Buscar dicotómicamente su lugar en las posiciones anteriores
  - Mover hacia la derecha los restantes
  - Insertarlo

```
PROCEDURE inserbin(VAR a: Tipoarray);
   VAR
      i,j,m,L,R: Tipo_indice;x: Tipo_datos;
BEGIN
   FOR i:=2 TO n DO
        x:=a[i]; (*L:indice al elemento de la parte izquierda del subarray en el que
                  se realiza la búsqueda en cada paso*)
        L:=1;    (*R:índice al elemento de la parte derecha del subarray en el que
                  se realiza la búsqueda en cada paso*)
        R:=i;
        WHILE L<R DO
           m:=(L+R)DIV 2;
           IF a[m]<=x THEN
              L:=m+1


            ELSE
               R:=m
            END
         END;
         FOR j:=i TO R+1 BY -1 DO
            a[j]:=a[j-1](* desplazamiento *)
         END;
         a[R]:=x; (* inserción *)
    END
 END
```

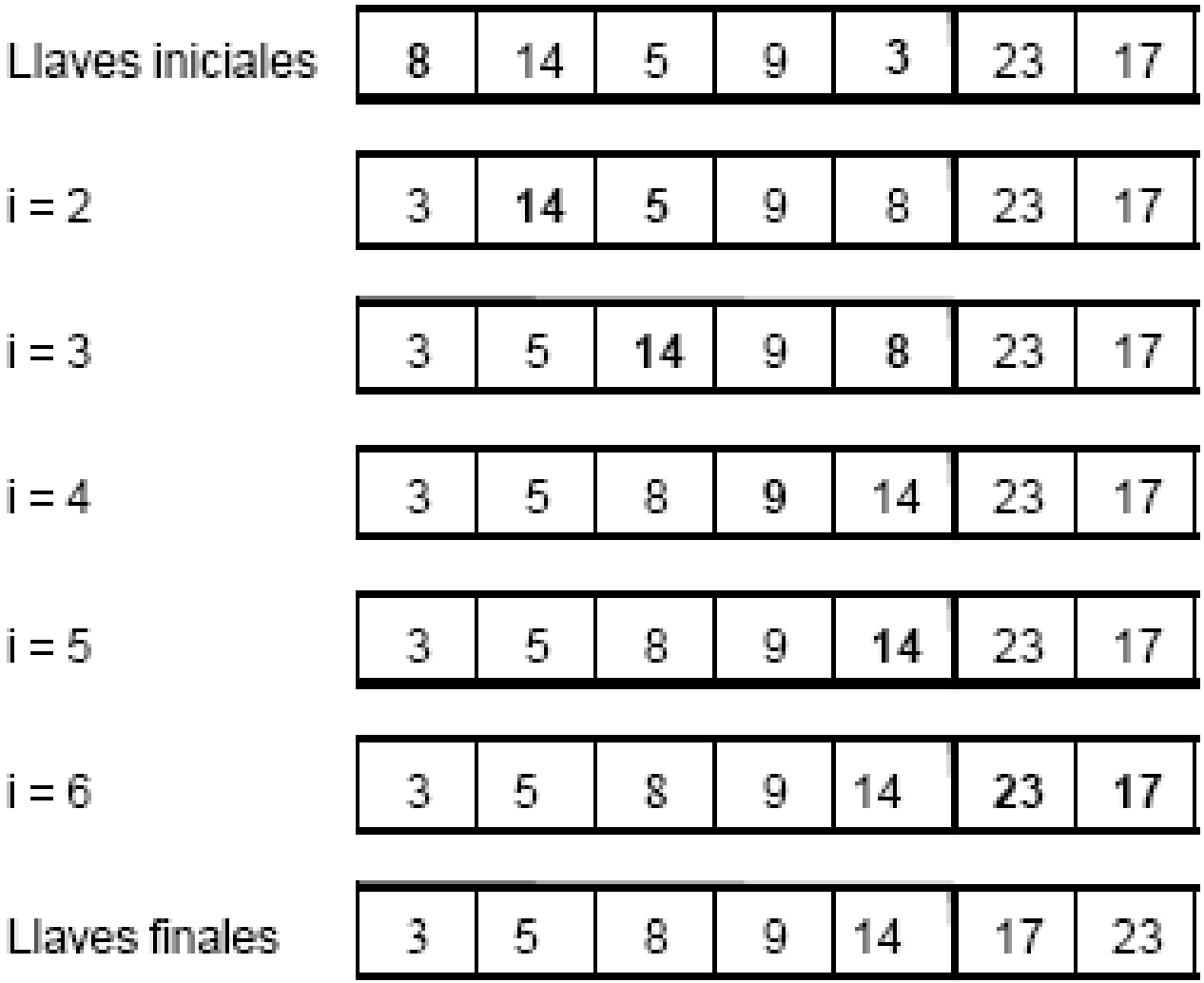
```
BEGIN
    FOR i:=2 to N DO
        X:= a[i];
        L:= 1;
        R:=i;
        WHILE L<R DO
            m:=(L+R) DIV 2;
            IF a[m] <= x THEN
                L:= m+1
              ELSE
                R:= m END
                END;
        FOR J:= i TO R+1 BY -1 DO
            a[j]:= a[j-1] (*desplazamiento*)
        END;
        a[R]:=x  (*inserción*)
    END
END
```

Algoritmo de clasificación por inserción binaria

2.2.2. Clasificación por selección directa.

- Al hacer un movimiento colocamos el elemento en su sitio definitivo.
- En un primer paso se recorre el arreglo hasta encontrar el elemento menor que se intercambia con el de la primera posición.
- Seguidamente se considera únicamente la parte del arreglo no ordenada y se repite el proceso:
  - Seleccionar el elemento menor de la parte del arreglo no ordenada
  - Colocarlo en la primera posición de la parte no ordenada del arreglo.

```
PROCEDURE selecdir(VAR a:Tipoarray);
   VAR
      i,j,k: Tipo_indice; x: Tipo_datos;
BEGIN
   FOR i:=1 TO n-1 DO
         k:=i;
         x:=a[i];          (* selección inicial *)
         FOR j:=i+1 TO n DO       (* búsqueda en secuencia fuente *)
            IF a[j]<x THEN
               k:=j;
               x:=a[k] (* selección *)
            END
         END;
         a[k]:=a[i];      (*intercambio*)
         a[i]:=x;
   END
END selecdir;
```

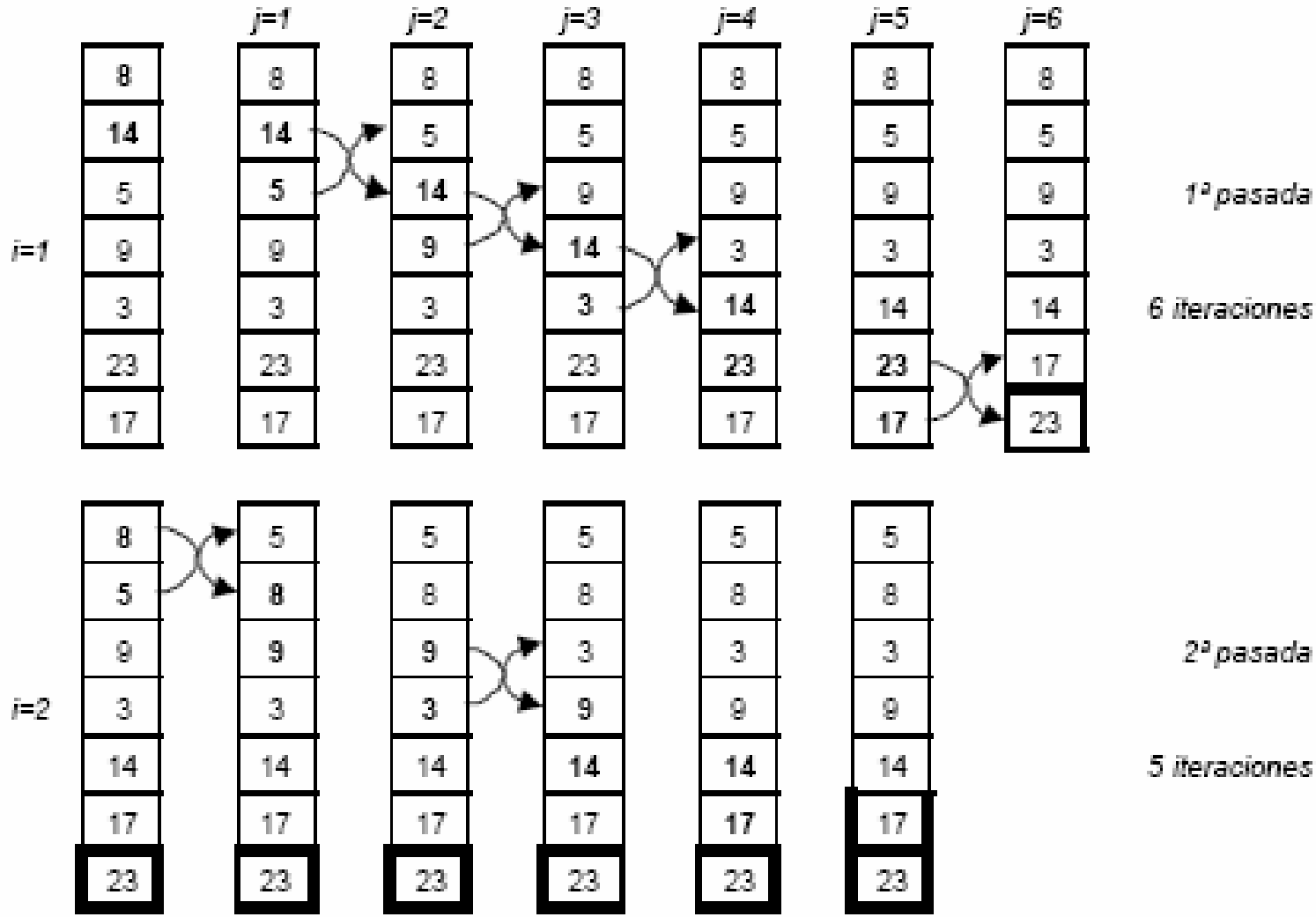

Llaves iniciales
8 14 5 9 3 23 17
i = 2
3 14 5 9 8 23 17
i = 3
3 5 14 9 8 23 17
i = 4
3 5 8 9 14 23 17
i = 5
3 5 8 9 14 23 17
i = 6
3 5 8 9 14 23 17
Llaves finales
3 5 8 9 14 17 23

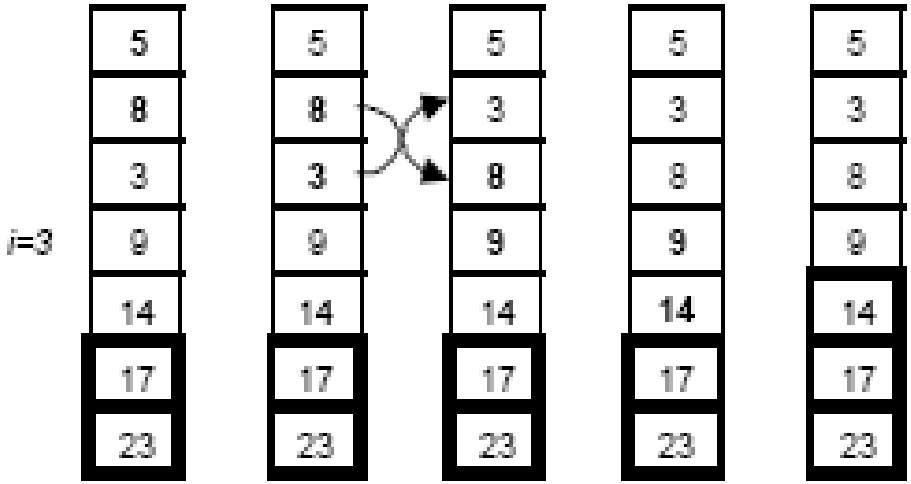
Comparación de la inserción con la selección

- Inserción directa:
  - En cada paso un elemento siguiente de la secuencia fuente y todos los del arreglo destino.
- Selección directa:
  - Todos los elementos del arreglo fuente para encontrar el que es menor y depositarlo como el siguiente elemento de la secuencia destino.
- Conclusión:
  - El método de selección directa es preferible al de inserción directa ya que su número de movimientos promedio es de orden n*Ln n.
  - En general, la implementación de una comparación tiene un coste inferior al de un movimiento.
  - Por tanto, la selección directa será la opción a elegir en general, aunque la inserción es algo más rápida cuando las llaves se clasifican predominantemente al principio.

2.2.3. Clasificación por intercambio:
Clasificación por intercambio directo burbuja

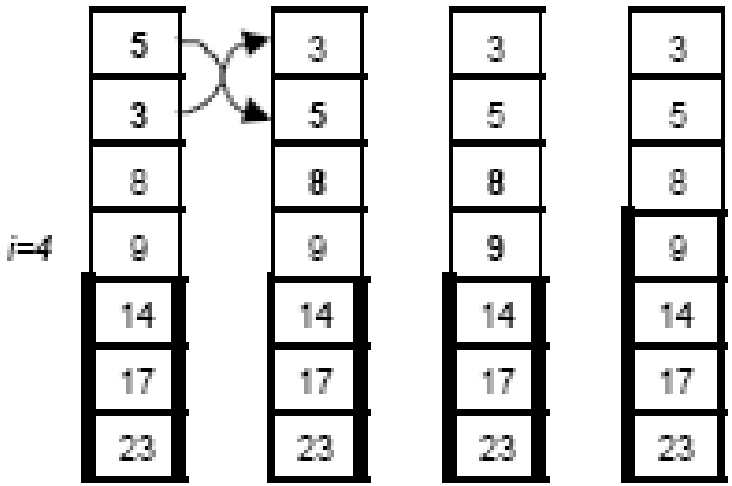
- Característica principal:
  - Intercambio de dos elementos.
- Método:
  - Comparar e intercambiar pares de elementos contiguos hasta clasificar todos los elementos.


j=1
j=2
j=3
j=4
j=5
j=6
i=1
1ª pasada
6 iteraciones
i=2
2ª pasada
5 iteraciones



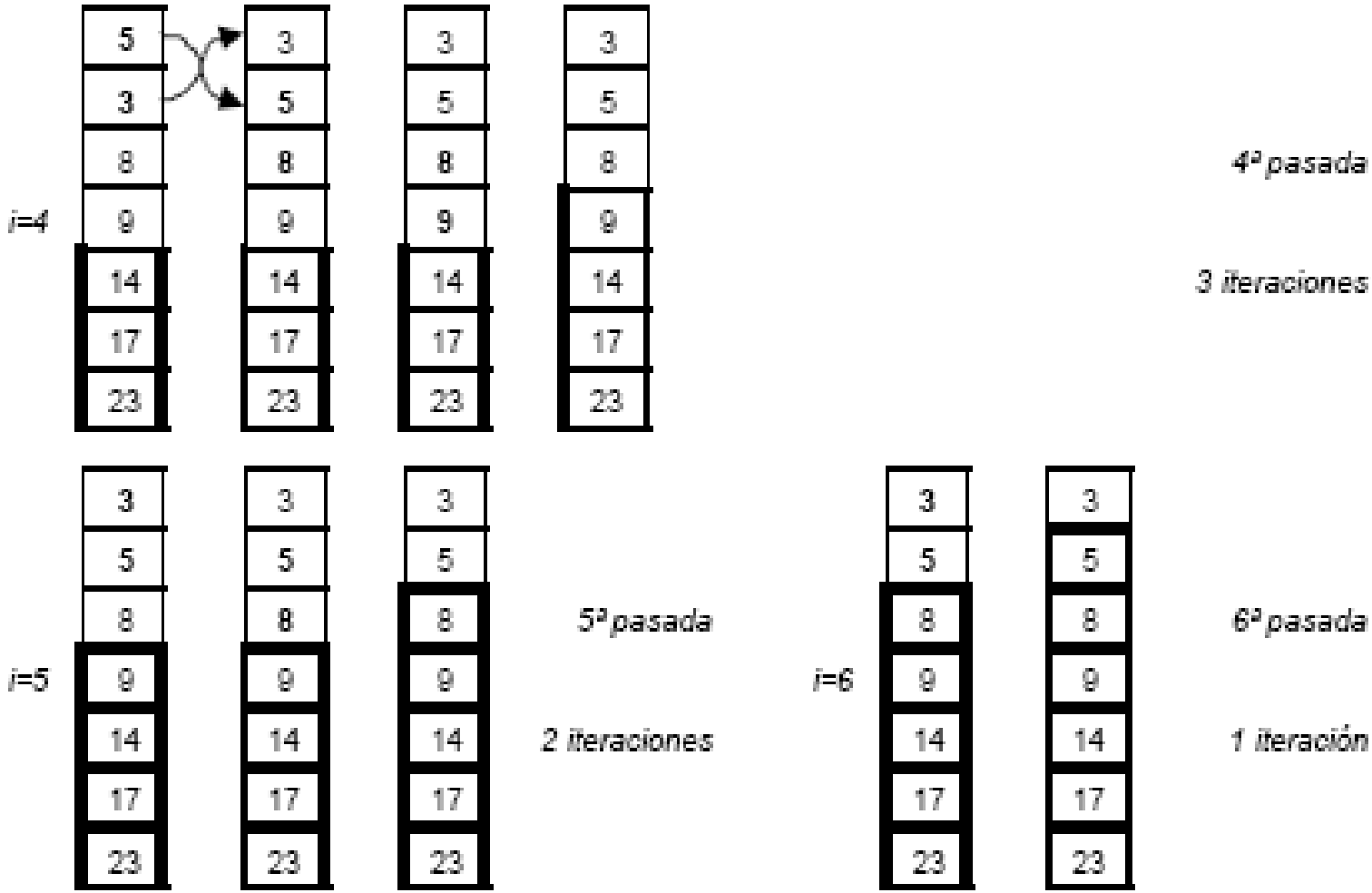
3ª pasada

4 iteraciones



4ª pasada

3 iteraciones

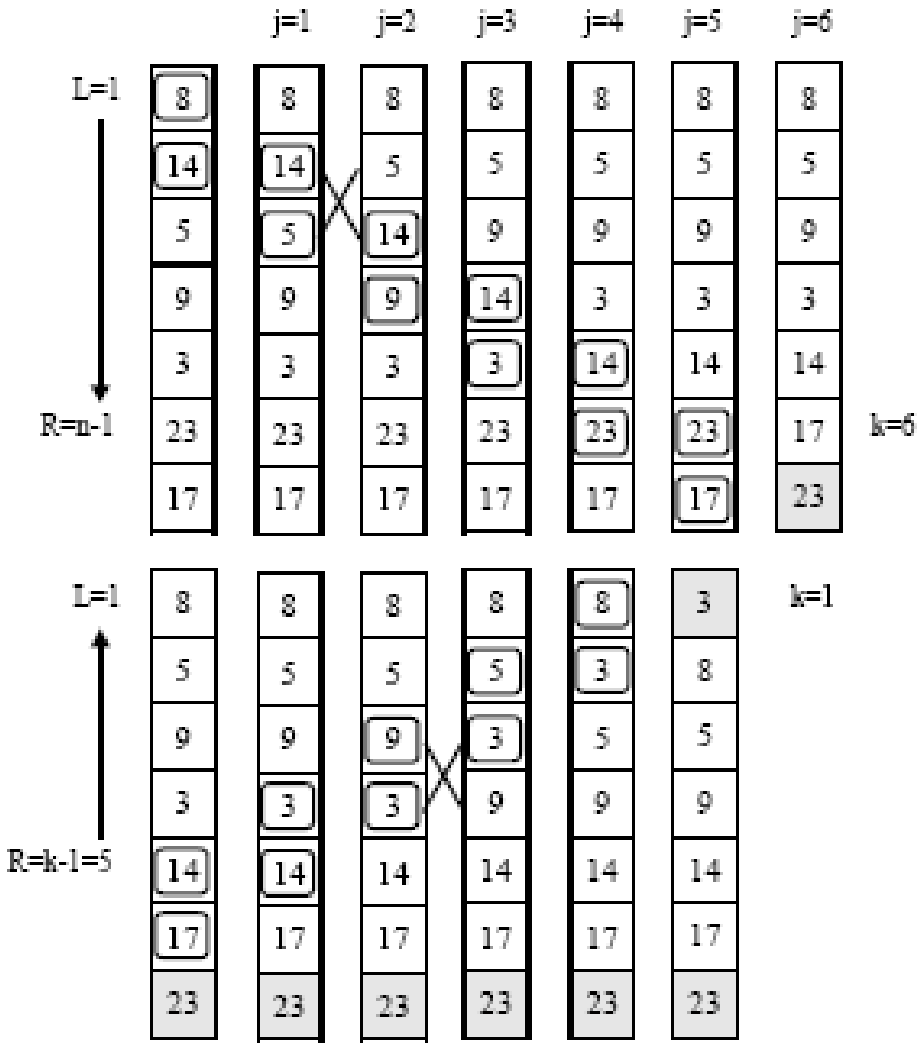

i=4
5 3 8 9 14 17 23
3 5 8 9 14 17 23
3 5 8 9 14 17 23
3 5 8 9 14 17 23
4ª pasada
3 iteraciones
i=5
3 5 8 9 14 17 23
3 5 8 9 14 17 23
3 5 8 9 14 17 23
5ª pasada
2 iteraciones
i=6
3 5 8 9 14 17 23
3 5 8 9 14 17 23
6ª pasada
1 iteración

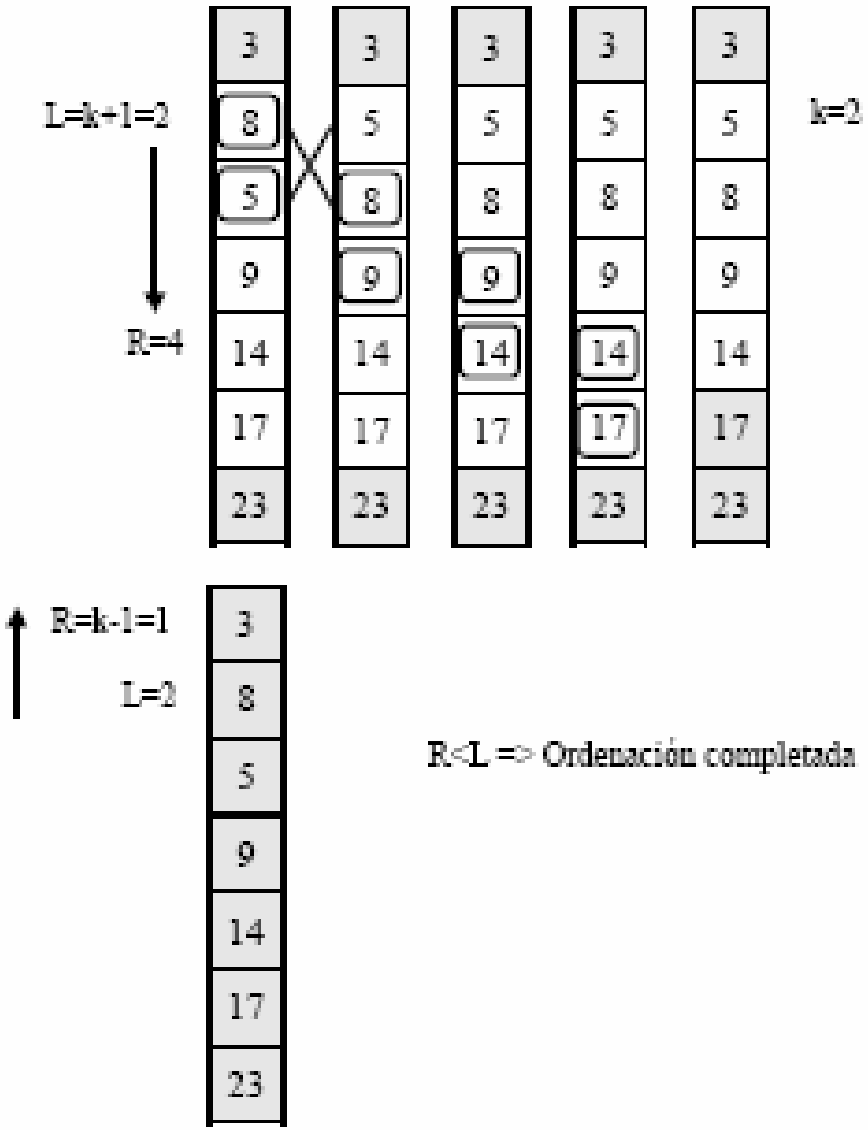
```
PROCEDURE burbuja(VAR a: Tipoarray);
   VAR
      i,j: Tipo_indice; aux: Tipo_datos;
BEGIN
   FOR i:=1 TO n-1 DO
        FOR j:=1 TO n-i DO
           IF a[j]>a[j+1] THEN   (*comparar elementos contiguos*)
              aux:=a[j+1];         (*intercambiar*)
              a[j+1]:=a[j];
              a[j]:=aux;
           END;
        END;
   END;
END burbuja;
```

Clasificación por sacudida o vibración

- Mejora el método de la clasificación por burbuja alternando la dirección de pases consecutivos.
- El segundo pase comienza en las posiciones finales y se recorre en sentido inverso al anterior.
- Así los pases impares se hacen en sentido descendente y los pares en sentido contrario.
- La parte del arreglo a ordenar estará en todo momento marcada por los índices L y R.
- En cada pasada hacia el final se decrementa el índice final (R) y en cada pasada hacia el inicio se incrementa el índice de inicio (L).
- El algoritmo se para, obviamente, cuando los índices se cruzan.
- Análisis:
  - No se mejora el número de movimientos a realizar.
  - La mejora se realiza únicamente en el número de comparaciones.

Clasificación por sacudida o por vibración




L=k+1=2
R=4
k=2
R=k-1=1
L=2
R<L => Ordenación completada

Fórmulas analíticas para los métodos de clasificación directa:

	Mín	Prom	Máx
C Inserción Directa M	(n-1) 2(n-1)	$(n^2+n-2)/4$ $(n^2+9n-10)/4$	$(n^2+n-2)/2$ $(n^2+3n-4)/2$
C Inserción Binaria M	n lgn n-1	n lgn $(n^2+7n-8)/4$	n lgn $(n^2+3n-4)/2$
C Selección Directa M	$(n^2-n)/2$ 3(n-1)	$(n^2-n)/2$ n[ln(n)+0.57]	$(n^2-n)/2$ $n^2/4+3(n-1)$
C Intercambio Directo M	$(n^2-n)/2$ 0	$(n^2-n)/2$ $3(n^2-n)/4$	$(n^2-n)/2$ $3*(n^2-n)/2$
C Por Vibración M	(n-1) 0	$½[n^2-n(k_2+lnn)]$ $O(n^2)$	$O(n^2)$ $O(n^2)$

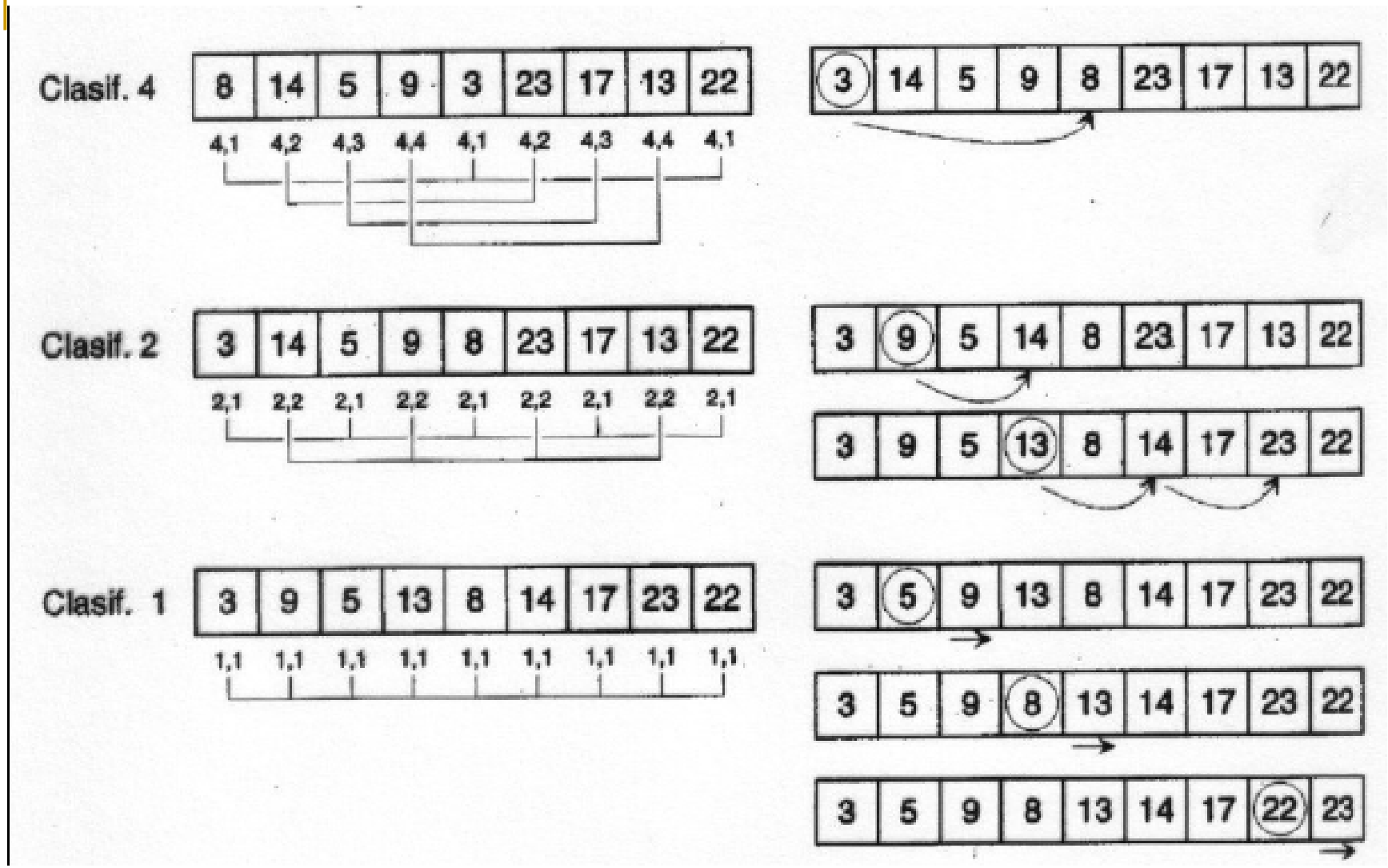
Conclusiones

- La complejidad de los métodos directos de clasificación de arreglos es del $O(n^2)$.
- Veremos que para los métodos de clasificación avanzada la complejidad es del O(n*lg(n)).
- El método de selección directa es preferible al de inserción directa ya que su número de movimientos promedio es de orden n*Lg (n).
- En general, la implementación de una comparación tiene un coste inferior al de un movimiento.
- Por tanto, la selección directa será la opción a elegir en general, aunque la inserción es algo más rápida cuando las llaves se clasifican predominantemente al principio

2.3. Métodos de clasificación avanzados

Inserción por incremento decreciente (Shell)

- Es una mejora del de inserción directa.
- Este método propuesto por D .L, Shell
- Se basa en la ordenación por inserción de los elementos que difieren, h_1 posiciones, después de los que difieren h_2 y así sucesivamente según un conjunto de incrementos decrecientes h_i, hasta ordenar los que difieren en h_t,=1.
- Existen numerosos conjuntos de incrementos decrecientes que garantizan la clasificación del arreglo, como por ejemplo: 1,2,4, .,.; 1,3,5,9, ...; 1,4, 13,40, 121 ,,,.; 1,3,7, I5,31, ...; etc,
- Se comienza en cada caso por incremento mayor que permita el tamaño del arreglo y se va reduciendo en cada actuación.


Clasif. 4
8 14 5 9 3 23 17 13 22
4,1 4,2 4,3 4,4 4,1 4,2 4,3 4,4 4,1
3 14 5 9 8 23 17 13 22
Clasif. 2
3 14 5 9 8 23 17 13 22
2,1 2,2 2,1 2,2 2,1 2,2 2,1 2,2 2,1
3 9 5 14 8 23 17 13 22
3 9 5 13 8 14 17 23 22
Clasif. 1
3 9 5 13 8 14 17 23 22
1,1 1,1 1,1 1,1 1,1 1,1 1,1 1,1 1,1
3 5 9 13 8 14 17 23 22
3 5 9 8 13 14 17 23 22
3 5 9 8 13 14 17 22 23

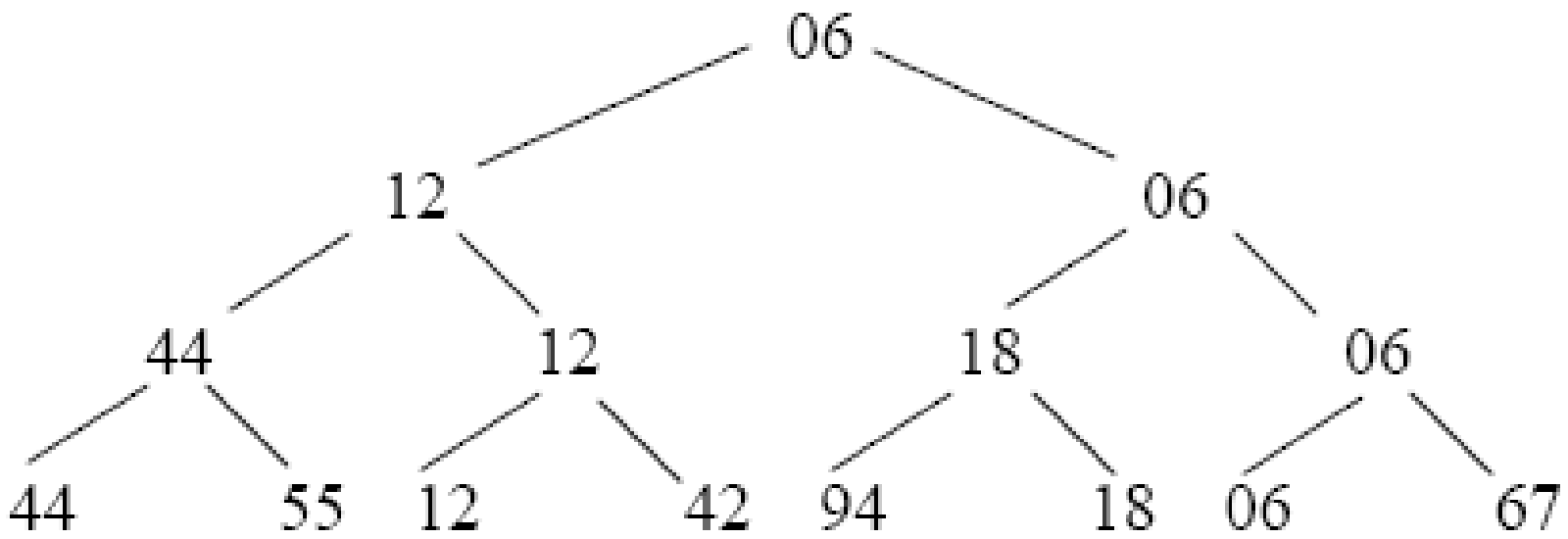
```
PROCEDURE shell(VAR a: Tipoarray);
CONST
   t=3;       (*número de ordenaciones por inserción directa*)
VAR
   i,j,k,
   s: Tipo_indice;    (*posición del centinela de cada clasificación*)
   m:[1..t];           (*índice que indica el número de ordenación*)
   h:ARRAY[1..t] OF INTEGER;  (*vector de incrementos*)

BEGIN
     h[1]:=4; h[2]:=2; h[3]:=1;
     FOR m:=1 TO t DO
         k:=h[m]; s:=-k;
         FOR i:= k+1 TO n DO
            j:=i-k;
            IF s=0 THEN s:=-k END;
            s:=s+1; a[s]:=a[i];          (*Inserción directa*)
            WHILE a[s]<a[j] DO
               a[j+k]:= a[j]; j:=j-k
            END;
            a[j+k]:=a[s]
         END;
     END
END shell;
```

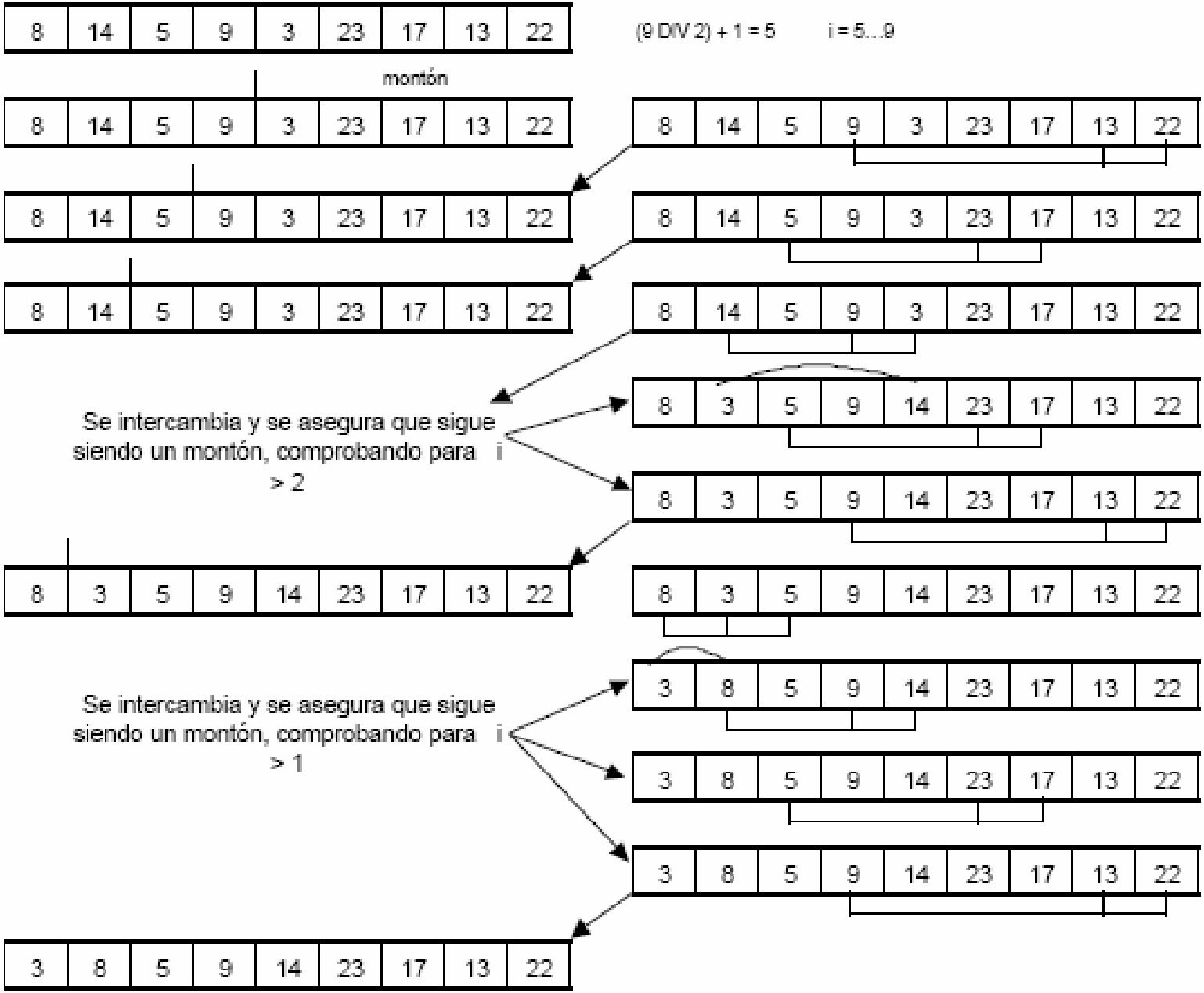
2.3.2. Clasificación por montón (Floyd)

- Un montículo o montón se define como una secuencia de llaves h_i , i = L, L+1,...R tal que mantienen la siguiente relación de orden:
  - $h_i <= h_{2i}$ y $h_i <= h_{2i+1}$ para i = L,...,R/2
- Dado un arreglo de n elementos, $h_1...h_n$, los elementos a partir de la mitad del arreglo, k=(nDIV2)+1, hasta el final, forman un montón ($h_k...h_n$) puesto que no hay dos índices i,j en esta parte que no satisfagan la definición de montón.
- Seguidamente se amplía ($h_k...h_n$) tomando un elemento por la izquierda y se reconstruye el montón con este nuevo elemento y así sucesivamente.

- Mejora el método de clasificación por selección directa.

1°) *Construcción del árbol*




8 14 5 9 3 23 17 13 22
(9 DIV 2) + 1 = 5
i = 5...9
montón
Se intercambia y se asegura que sigue siendo un montón, comprobando para i > 2
Se intercambia y se asegura que sigue siendo un montón, comprobando para i > 1
3 8 5 9 14 23 17 13 22

2.3.3. Clasificación por partición (clasif. rápida)

- Se basa en el hecho de que cuanto mayor sea la distancia entre los elementos que se intercambian más eficaz será la clasificación.
- Se elige un valor de llave al azar, x, denominado pivote. Se recorre el arreglo desde la izquierda hasta encontrar una llave mayor que el pivote, y desde la derecha hasta encontrar una menor
- Se intercambian y se repite el proceso hasta que los índices de incremento y decremento se crucen.
- De esta forma tendremos a la izquierda todos los elementos menores que el pivote y a la derecha los mayores.
- A cada una de estas partes se denominan particiones.

Clasificación por partición (Clasificación rápida)

Se basa en el método de clasificación por intercambio.

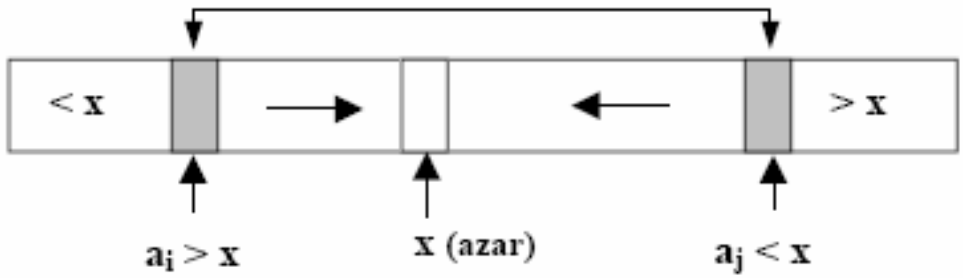
Método:

1º) Se selecciona un elemento al azar x.

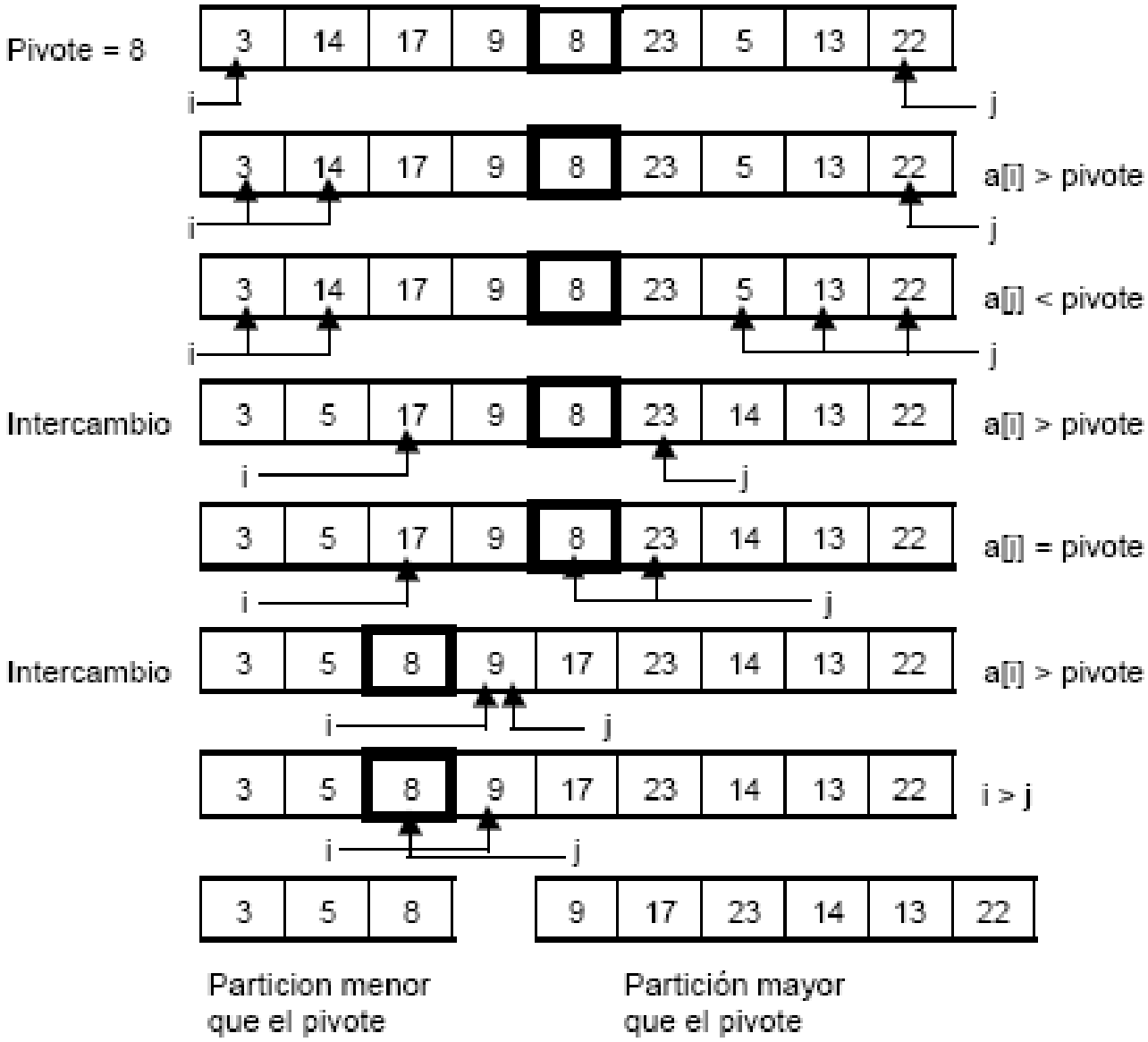
2º) Se rastrea desde la izquierda hasta un a_i tal que $a_i > x$.

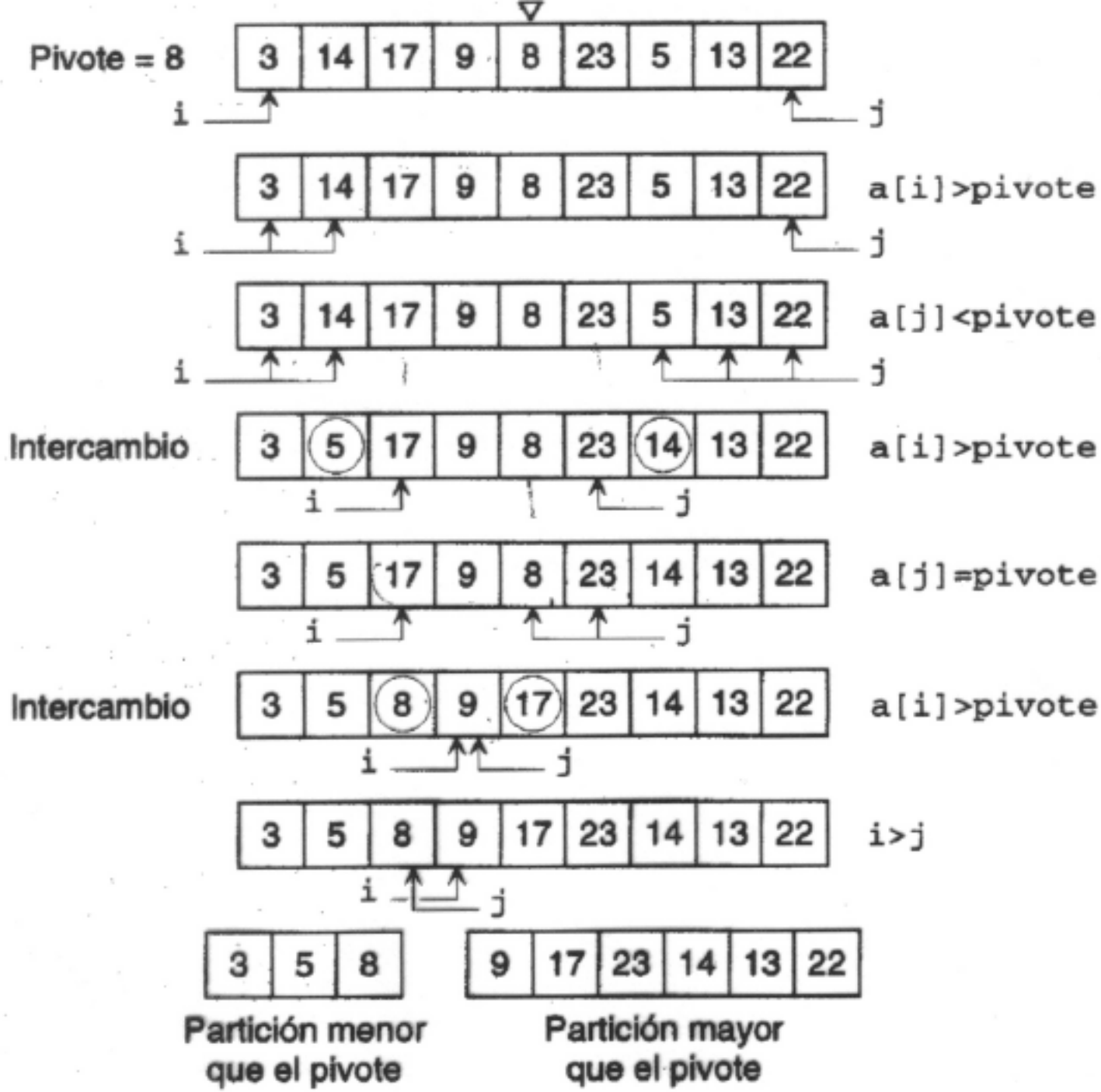
3º) Se rastrea desde la derecha hasta un a_j tal que $a_j < x$.

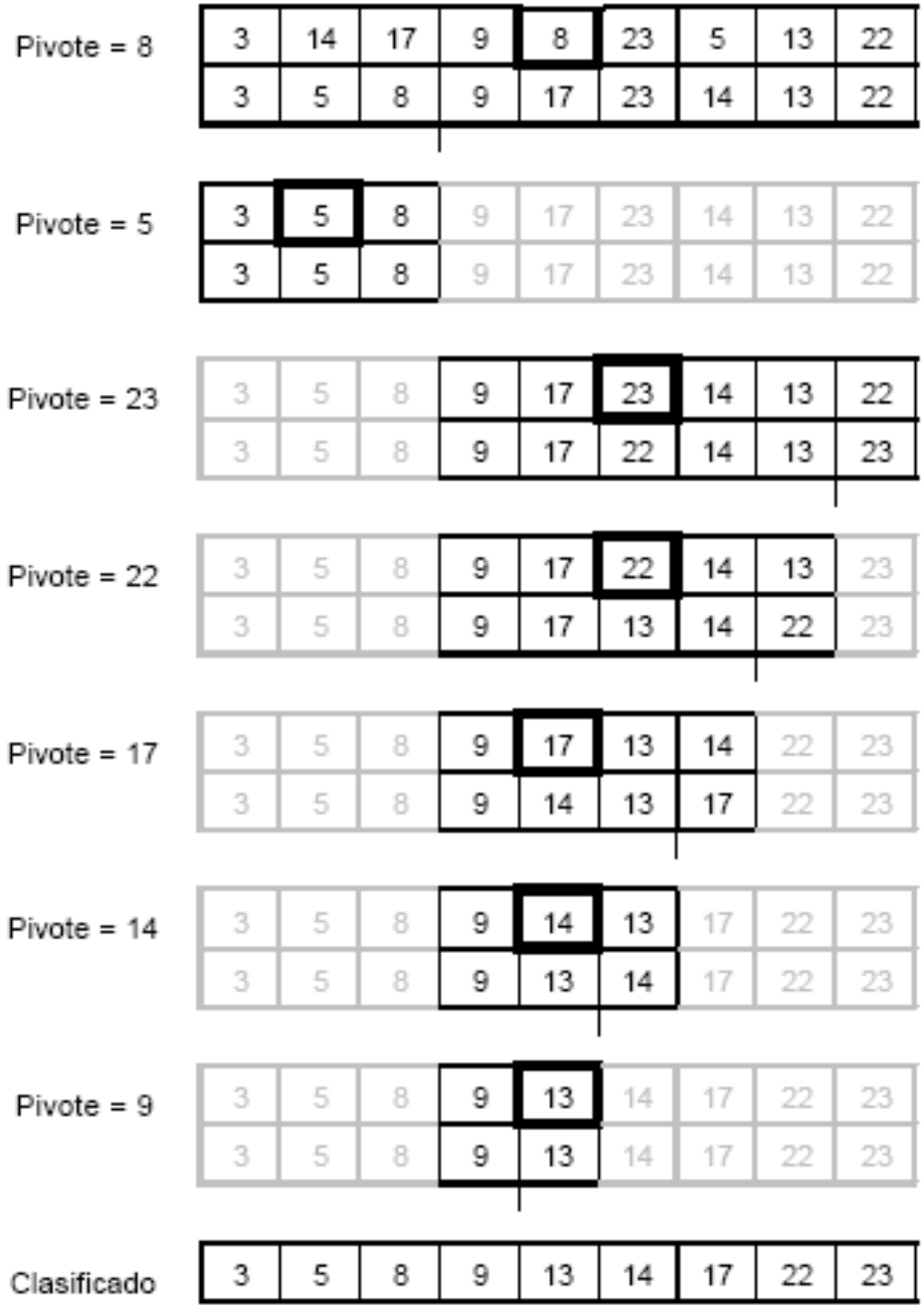
4º) Se intercambian (i=i+1, j=j-1).

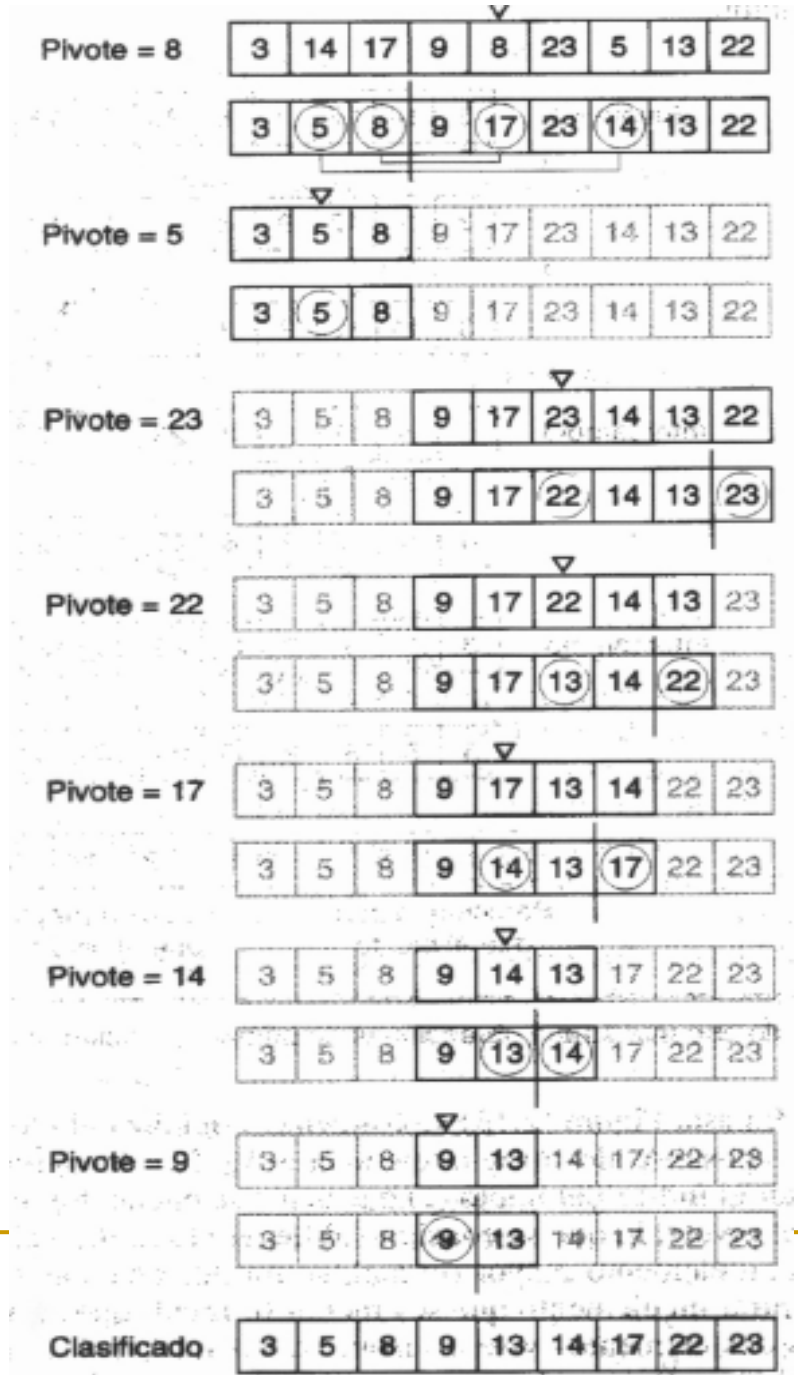


5º) Se repiten estos pasos hasta que i>j.


Pivote = 8
3 14 17 9 8 23 5 13 22
i
j
3 14 17 9 8 23 5 13 22
a[i] > pivote
i
j
3 14 17 9 8 23 5 13 22
a[j] < pivote
i
j
Intercambio
3 5 17 9 8 23 14 13 22
a[i] > pivote
i
j
3 5 17 9 8 23 14 13 22
a[j] = pivote
i
j
Intercambio
3 5 8 9 17 23 14 13 22
a[i] > pivote
i
j
3 5 8 9 17 23 14 13 22
i > j
i
j
3 5 8
9 17 23 14 13 22
Particion menor que el pivote
Partición mayor que el pivote


Pivote = 8
3 14 17 9 8 23 5 13 22
i
j
3 14 17 9 8 23 5 13 22
a[i]>pivote
i
j
3 14 17 9 8 23 5 13 22
a[j]<pivote
i
j
Intercambio
3 5 17 9 8 23 14 13 22
a[i]>pivote
i
j
3 5 17 9 8 23 14 13 22
a[j]=pivote
i
j
Intercambio
3 5 8 9 17 23 14 13 22
a[i]>pivote
i
j
3 5 8 9 17 23 14 13 22
i>j
i
j
3 5 8
9 17 23 14 13 22
Partición menor que el pivote
Partición mayor que el pivote


Pivote = 8
3 14 17 9 8 23 5 13 22
3 5 8 9 17 23 14 13 22
Pivote = 5
3 5 8 9 17 23 14 13 22
3 5 8 9 17 23 14 13 22
Pivote = 23
3 5 8 9 17 23 14 13 22
3 5 8 9 17 22 14 13 23
Pivote = 22
3 5 8 9 17 22 14 13 23
3 5 8 9 17 13 14 22 23
Pivote = 17
3 5 8 9 17 13 14 22 23
3 5 8 9 14 13 17 22 23
Pivote = 14
3 5 8 9 14 13 17 22 23
3 5 8 9 13 14 17 22 23
Pivote = 9
3 5 8 9 13 14 17 22 23
3 5 8 9 13 14 17 22 23
Clasificado
3 5 8 9 13 14 17 22 23


Pivote = 8
3 14 17 9 8 23 5 13 22
3 5 8 9 17 23 14 13 22
Pivote = 5
3 5 8 9 17 23 14 13 22
3 5 8 9 17 23 14 13 22
Pivote = 23
3 5 8 9 17 23 14 13 22
3 5 8 9 17 22 14 13 23
Pivote = 22
3 5 8 9 17 22 14 13 23
3 5 8 9 17 13 14 22 23
Pivote = 17
3 5 8 9 17 13 14 22 23
3 5 8 9 14 13 17 22 23
Pivote = 14
3 5 8 9 14 13 17 22 23
3 5 8 9 13 14 17 22 23
Pivote = 9
3 5 8 9 13 14 17 22 23
3 5 8 9 13 14 17 22 23
Clasificado
3 5 8 9 13 14 17 22 23

.11 Clasificación por partición (quicksort).

```
PROCEDURE particion;
   PROCEDURE ordena (L, R : Tipo_indice);
   VAR
      i,j     : Tipo_indice;
      piv,aux : Tipo_datos;
   BEGIN
      i:=L; j:=R;
      piv:=a[(L+R)DIV 2];        (* pivote *)
      REPEAT
         WHILE a[i] < piv DO i := i+1 END;
         WHILE piv < a[j] DO j := j-1 END;
         IF i <= j THEN
            aux:=a[i]; a[i]:=a[j]; a[j]:=aux;    (*intercambio*)
            i:=i+1;
            j:=j-1
         END
      UNTIL i>j;
      IF L<j THEN ordena(L,j) END;
      IF i<R THEN ordena(i,R) END;
   END ordena;
BEGIN
   ordena (1,n);
END particion;
```

Obtención del k-ésimo menor elemento.

- La clasificación por partición permite, además de clasificar un arreglo, calcular de forma eficaz el k-ésimo elemento mayor del mismo.
- La mediana consiste en calcular el k-ésimo mayor, con k=n/2.

Comparación (i)

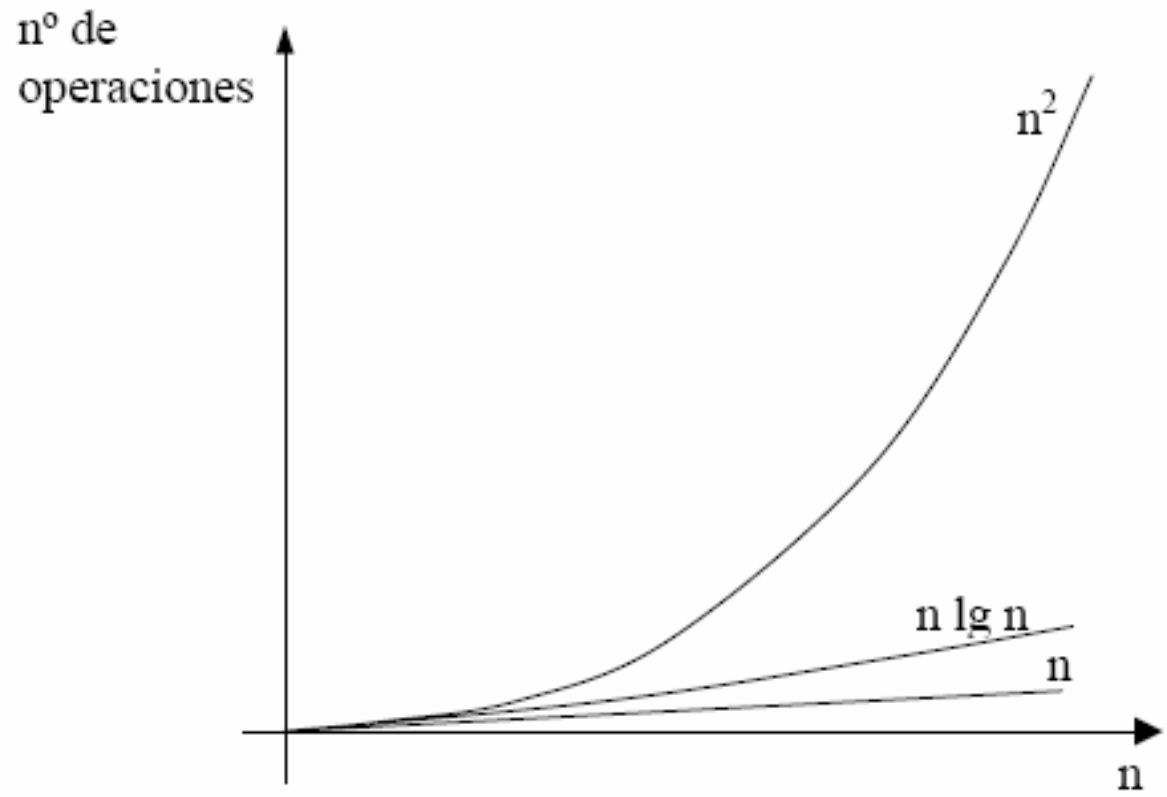
- Comparando los métodos directos de clasificación sobre arreglos,
  - La selección directa será la opción a elegir en general,
  - La inserción es algo más rápida cuando las llaves predominantemente se clasifican al principio y
  - La vibración puede ser mejor cuando el arreglo está inicialmente casi ordenado.
- Respecto a los métodos avanzados
  - La clasificación rápida tiene un magnífico comportamiento en los casos mejor y promedio, sin embargo en el caso peor es deficiente (cuando las llaves son muy parecidas en valor).
  - En este caso es aconsejable otro método, por ejemplo, la clasificación por montón.

Comparación (ii)

- Los métodos avanzados
  - Presentan unos costes computacionales claramente inferiores a los directos pero no son los más eficaces para todo *n* ya que requieren más operaciones auxiliares que los directos.
- Para *n* pequeño
  - Son preferibles los **métodos directos** ya que en estos casos las diferencias entre los órdenes n^2 y n*log(n) no son muy significativas y las operaciones auxiliares de los avanzados hacen que estos métodos sean menos eficaces.
- En el caso de clasificación por partición,
  - En principio se realizará el proceso de partición, pero sólo hasta que el tamaño de las particiones es lo suficientemente pequeño para aplicar un método directo eficaz, como la selección directa.

Comparación entre métodos directos y avanzados

- Los métodos avanzados requieren más operaciones que los directos:
  - los primeros sólo recomendables para *n* grandes y los segundos para *n* pequeños.



n	**$n\lg_2 n$**	**n^2**
10	34	100
50	283	2.500
1.000	≈10.000	1 millón
10^6	≈20 millones	1 billón