Algoritmos de ordenación

Cuestiones generales

Su finalidad es organizar ciertos datos (normalmente arreglos o archivos) en un orden creciente o decreciente mediante una regla prefijada (numérica, alfabética...). Atendiendo al tipo de elemento que se quiera ordenar puede ser:

Ordenación interna: Los datos se encuentran en memoria (ya sean arreglos, listas, etc.), y son de acceso aleatorio o directo (se puede acceder a un determinado campo sin pasar por los anteriores).

Ordenación externa: Los datos están en un dispositivo de almacenamiento externo (archivos), y su ordenación es más lenta que la interna.

Ordenación interna

Los métodos de ordenación interna se aplican principalmente a arreglos unidimensionales. Los principales algoritmos de ordenación interna son:

Selección: Este método consiste en buscar el elemento más pequeño del arreglo y ponerlo en primera posición; luego, entre los restantes, se busca el elemento más pequeño y se coloca en segundo lugar, y así sucesivamente hasta colocar el último elemento. Por ejemplo, si tenemos el arreglo {40,21,4,9,10,35}, los pasos a seguir son:

```
{4,21,40,9,10,35} <-- Se coloca el 4, el más pequeño, en primera posición
: se cambia el 4 por el 40.
{4,9,40,21,10,35} <-- Se coloca el 9, en segunda posición: se cambia el 9
por el 21.
{4,9,10,21,40,35} <-- Se coloca el 10, en tercera posición: se cambia el
10 por el 40.
{4,9,10,21,40,35} <-- Se coloca el 21, en tercera posición: ya está
colocado.
{4,9,10,21,35,40} <-- Se coloca el 35, en tercera posición: se cambia el
35 por el 40.
```

Burbuja: Consiste en comparar pares de elementos adyacentes e intercambiarlos entre sí hasta que estén todos ordenados. Con el arreglo anterior, {40,21,4,9,10,35}:

```
Primera pasada:
{21,40,4,9,10,35} <-- Se cambia el 21 por el 40.
{21,4,40,9,10,35} <-- Se cambia el 40 por el 4.
{21,4,9,40,10,35} <-- Se cambia el 9 por el 40.
{21,4,9,10,40,35} <-- Se cambia el 40 por el 10.
{21,4,9,10,35,40} <-- Se cambia el 35 por el 40.

Segunda pasada:
{4,21,9,10,35,40} <-- Se cambia el 21 por el 4.
```

```
{4,9,21,10,35,40} <-- Se cambia el 9 por el 21.
{4,9,10,21,35,40} <-- Se cambia el 21 por el 10.
```

Ya están ordenados, pero para comprobarlo habría que acabar esta segunda comprobación y hacer una tercera.

Inserción directa: En este método lo que se hace es tener una sublista ordenada de elementos del arreglo e ir insertando el resto en el lugar adecuado para que la sublista no pierda el orden. La sublista ordenada se va haciendo cada vez mayor, de modo que al final la lista entera queda ordenada. Para el ejemplo {40,21,4,9,10,35}, se tiene:

```
{40,21,4,9,10,35}  <-- La primera sublista ordenada es {40}.

Insertamos el 21:
{40,40,4,9,10,35}  <-- aux=21;
{21,40,4,9,10,35}  <-- Ahora la sublista ordenada es {21,40}.

Insertamos el 4:
{21,40,40,9,10,35} <-- aux=4;
{21,21,40,9,10,35} <-- aux=4;
{4,21,40,9,10,35}  <-- Ahora la sublista ordenada es {4,21,40}.

Insertamos el 9:
{4,21,40,40,10,35} <-- aux=9;
{4,21,21,40,10,35} <-- aux=9;
{4,9,21,40,10,35}  <-- Ahora la sublista ordenada es {4,9,21,40}.

Insertamos el 10:
{4,9,21,40,40,35}  <-- aux=10;
{4,9,21,21,40,35}  <-- aux=10;
{4,9,10,21,40,35}  <-- Ahora la sublista ordenada es {4,9,10,21,40}.

Y por último insertamos el 35:
{4,9,10,21,40,40}  <-- aux=35;
{4,9,10,21,35,40}  <-- El arreglo está ordenado.
```

Shell: Es una mejora del método de inserción directa, utilizado cuando el arreglo tiene un gran número de elementos. En este método no se compara a cada elemento con el de su izquierda, como en el de inserción, sino con el que está a un cierto número de lugares (llamado salto) a su izquierda. Este salto es constante, y su valor inicial es N/2 (siendo N el número de elementos, y siendo división entera). Se van dando pasadas hasta que en una pasada no se intercambie ningún elemento de sitio. Entonces el salto se reduce a la mitad, y se vuelven a dar pasadas hasta que no se intercambie ningún elemento, y así sucesivamente hasta que el salto vale 1.

Por ejemplo, lo pasos para ordenar el arreglo {40,21,4,9,10,35} mediante el método de Shell serían:

```
Salto=3:
   Primera pasada:
   {9,21,4,40,10,35} <-- se intercambian el 40 y el 9.
   {9,10,4,40,21,35} <-- se intercambian el 21 y el 10.
Salto=1:
   Primera pasada:
```

```
{9,4,10,40,21,35} <-- se intercambian el 10 y el 4.
{9,4,10,21,40,35} <-- se intercambian el 40 y el 21.
{9,4,10,21,35,40} <-- se intercambian el 35 y el 40.
Segunda pasada:
{4,9,10,21,35,40} <-- se intercambian el 4 y el 9.
```

Con sólo 6 intercambios se ha ordenado el arreglo, cuando por inserción se necesitaban muchos más.

Intercalación: no es propiamente un método de ordenación, consiste en la unión de dos arreglos ordenados de modo que la unión esté también ordenada. Para ello, basta con recorrer los arreglos de izquierda a derecha e ir cogiendo el menor de los dos elementos, de forma que sólo aumenta el contador del arreglo del que sale el elemento siguiente para el arreglo-suma. Si quisiéramos sumar los arreglos {1,2,4} y {3,5,6}, los pasos serían:

```
Inicialmente: i1=0, i2=0, is=0.
Primer elemento: mínimo entre 1 y 3 = 1. Suma={1}. i1=1, i2=0, is=1.
Segundo elemento: mínimo entre 2 y 3 = 2. Suma={1,2}. i1=2, i2=0, is=2.
Tercer elemento: mínimo entre 4 y 3 = 3. Suma={1,2,3}. i1=2, i2=1, is=3.
Cuarto elemento: mínimo entre 4 y 5 = 4. Suma={1,2,3,4}. i1=3, i2=1,
is=4.
Como no quedan elementos del primer arreglo, basta con poner los
elementos que quedan del segundo arreglo en la suma:
Suma={1,2,3,4}+{5,6}={1,2,3,4,5,6}
```

Mergesort: Una técnica muy poderosa para el diseño de algoritmos es "*Dividir para conquistar*". Los algoritmos de este tipo se caracterizan por estar diseñados siguiendo estrictamente las siguientes fases:

- Dividir: Se divide el problema en partes más pequeñas.
- Conquistar: Se resuelven recursivamente los problemas más chicos.
- Combinar: Los problemas mas chicos de combinan para resolver el grande.

Los algoritmos que utilizan este principio son en la mayoría de los casos netamente recursivos como es el caso de mergesort**.**

El algoritmo de Mergesort es un ejemplo clásico de algoritmo que utiliza el principio de dividir para conquistar. Si el vector tiene más de dos elementos se lo divide en dos mitades, se invoca recursivamente al algoritmo y luego se hace una *intercalación* de las dos mitades ordenadas. Para el ejemplo {40,21,4,9,10,35}, se tiene:

```
{40,21,4},{9,10,35}         <-- Dividimos el arreglo en dos
{40},{21,4},{9,10,35}       <-- Dividimos el primer arreglo nuevamente
{40},{21},{4},{9,10,35}     <-- {40} ya está ordenado así que dividimos
{21,4}
{40},{4,21},{9,10,35}       <-- {21} y {4} están ordenados así que se
intercalan
{4,21,40},{9,10,35}         <-- se intercala {40} con {4,21}
{4,21,40},{9},{10,35}       <-- Se divide el segundo arreglo en dos
{4,21,40},{9},{10},{35}     <-- Se divide el último arreglo en dos
{4,21,40},{9},{10,35}       <-- Intercalamos {10} y {35}
{4,21,40},{9,10,35}         <-- Intercalamos {9} y {10,35}
```

```
{4,9,10,21,35,40}           <-- Intercalamos los dos arreglos y obtenemos
el resultado final.
```

Ordenación rápida (quicksort): Este método se basa en la táctica "divide y vencerás", que consiste en ir subdividiendo el arreglo en arreglos más pequeños, y ordenar éstos. Para hacer esta división, se toma un valor del arreglo como pivote, y se mueven todos los elementos menores que este pivote a su izquierda, y los mayores a su derecha. A continuación se aplica el mismo método a cada una de las dos partes en las que queda dividido el arreglo.

Normalmente se toma como pivote el primer elemento de arreglo, y se realizan dos búsquedas: una de izquierda a derecha, buscando un elemento mayor que el pivote, y otra de derecha a izquierda, buscando un elemento menor que el pivote. Cuando se han encontrado los dos, se intercambian, y se sigue realizando la búsqueda hasta que las dos búsquedas se encuentran. Por ejemplo, para dividir el arreglo {21,40,4,9,10,35}, los pasos serían:

```
{21,40,4,9,10,35} <-- se toma como pivote el 21. La búsqueda de izquierda
a derecha encuentra el valor 40, mayor que pivote, y la búsqueda de
derecha a izquierda encuentra el valor 10, menor que el pivote. Se
intercambian:
{21,10,4,9,40,35} <-- Si seguimos la búsqueda, la primera encuentra el
valor 40, y la segunda el valor 9, pero ya se han cruzado, así que
paramos. Para terminar la división, se coloca el pivote en su lugar (en
el número encontrado por la segunda búsqueda, el 9, quedando:
{9,10,4,21,40,35} <-- Ahora tenemos dividido el arreglo en dos arreglos
más pequeños: el {9,10,4} y el {40,35}, y se repetiría el mismo proceso.
```