Introducción

En este capitulo se presentan los algoritmos básicos para la clasificación sobre arreglos. Esta clasificación se realizará siempre sobre memoria principal (ordenación interna). Cuando el tamaño del arreglo a ordenar ya no quepa en memoria principal, se deberá recurrir a una clasificación en memoria secundaria (ordenación externa).

Las colecciones de elementos a clasificar se almacenarán en arreglos, de manera que **cualquier elemento puede ser accedido en el mismo tiempo**, es decir, con el mismo costo. La definición de tipos y variables será, por tanto, en Modula-2:

```
TYPE Tipo_datos = RECORD
                        llave : Tipo_llave;
                        (* otros componentes *)
                  END

VAR a: ARRAY [1..n] OF Tipo_datos;
```

La taxonomía realizada en términos de métodos de clasificación directa y métodos avanzados se debe a que los primeros explotan las ideas más intuitivas (directas) y en general presentan un coste computacional de orden n^2 , mientras que los segundos reducen este coste utilizando ideas y técnicas más elaboradas.

Para la representación de los ejemplos siguientes se utilizará el tipo de datos *Entero* ya que presenta una relación de orden bien conocida. De este modo, la declaración de tipos corresponderá a:

```
TYPE Tipo_datos = Integer;
```

Inserción directa

En cada paso, comenzando con i=2 y aumentando en una unidad, el i-esimo elemento de la secuencia fuente se toma y se transfiere a la secuencia destino insertándolo en el lugar correspondiente.

Se aplica la técnica del centinela como condición de terminación. El algoritmo completo es:

```
MODULE InsercionDirecta;
FROM InOut IMPORT WriteInt, WriteLn, WriteString;
       TYPE tipoarray=ARRAY[0..9] OF INTEGER;

       VAR i,j: INTEGER; n: INTEGER;
       VAR a: tipoarray;
BEGIN
   (* carga del arreglo de prueba en orden inverso (peor caso) *)
    n:=9;
    FOR i:=1 to 9 DO
        a[i]:=10-i
    END;

   (* Algoritmo de clasificación ------------------------------*)
    FOR i:=2 TO n DO
        a[0]:=a[i]; (* Garantiza la salida del while *)
        j:=i;
        WHILE a[0]<a[j-1] DO
              a[j]:=a[j-1]; (* movimientos *)
              j:=j-1
        END;
        a[j]:=a[0] (* inserción *)
    END;
  (* Fin Algoritmo de clasificación -------------------------*)

   (*Visualizaci¢n por pantalla de pasos intermedios*)
        FOR i:=1 TO n DO
           WriteInt(a[i],4)
        END;
        WriteLn;
END InsercionDirecta.
```

Análisis

El número de comparaciones C en la i-ésima extracción es a lo sumo i-1 y como máximo 1; suponiendo que todas las permutaciones de n llaves sean igualmente probables (i/2 en promedio). El número M de movimientos (asignación de elementos) es C_i+2 (incluido el centinela). Por tanto, los números totales de comparaciones y movimientos son:

Comparaciones	Movimientos
$C_{min}=n-1$ llaves arreglo inicial, ordenadas	$M_{min}=2(n-1)$ llaves arreglo inicial, ordenadas
$C_{max}=\frac{(n^2+n-2)}{2}$ llaves arreglo inicial, completamente desordenadas	$M_{max}=\frac{(n^2+3n-4)}{2}$ llaves arreglo inicial, completamente desordenadas
$C_{prom}=\frac{(n^2+n-2)}{4}$ llaves arreglo inicial, aleatoriamente ordenadas	$M_{prom}=\frac{(n^2+9n-10)}{4}$ llaves arreglo inicial, aleatoriamente ordenadas

Inserción binaria

Este algoritmo se basa en el de inserción directa teniendo en cuenta que la secuencia destino donde debe insertarse el elemento ya esta ordenada. Partiendo de esta premisa, la elección obvia es una búsqueda binaria que prueba la secuencia destino en la mitad y continúa buscando hasta encontrar el punto de inserción. El algoritmo completo es:

```
MODULE InsercionBinaria;
FROM InOut IMPORT WriteInt, WriteLn, WriteString;
       TYPE tipoarray=ARRAY[0..9] OF INTEGER;

       VAR i,j,m,L,R: INTEGER; n: INTEGER; x: INTEGER;
       VAR a: tipoarray;
BEGIN
   (* carga del arreglo de prueba en orden inverso (peor caso) *)
    n:=9;
    FOR i:=1 to 9 DO
        a[i]:=10-i
    END;

   (* Algoritmo de clasificación ------------------------------*)
    FOR i:=2 TO n DO
        x:=a[i];
        (* L: índice 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;
   (* Fin Algoritmo de clasificación -------------------------*)

   (*Visualización por pantalla de pasos intermedios*)
        FOR i:=1 TO n DO
           WriteInt(a[i],4)
        END;
        WriteLn;
END InsercionBinaria.
```

Análisis

El número de movimientos no mejora significativamente con respecto al método de inserción directa y seguirán siendo de orden n^2 .
La posición de inserción se encuentra cuando L=R. En consecuencia, el intervalo de búsqueda debe ser, al final, de longitud 1; lo que supone dividir a la mitad el intervalo de longitud $N=\log n$ veces, donde **n** es el número de veces que se divide por 2. La inserción se producirá cuando $C=\sum_{i=1}^{n}[\log i]$, aproximado por la integral $C=\int_{1}^{n}(\log x)\,dx$ quedando después de operar como $C=n(\log n-\log e)+\log e$

El número de comparaciones es esencialmente independiente del orden inicial de los elementos. Sin embargo dado el valor truncador de la división el número de comparaciones puede ser hasta uno más de lo previsto. Esto es debido a que las posiciones de inserción en el extremo inferior se localizan en promedio más pronto que las del extremo superior, con lo que se favorecen los casos en que los elementos están fuera de orden al principio. El número menor corresponde a cuando es orden es inverso y el máximo cuando ya están en orden.
La mejora sobre la inserción directa es sólo en el número de comparaciones, no al de movimientos requeridos. Como mover elementos es más lento que compararlos, la mejora no es radical.

Selección directa

Se basa en los siguientes principios:

1- Seleccionar el elemento con llave menor.
2- Intercambiarlo con el primer elemento.
3- Repetir las operaciones con $n-1$, luego con $n-2$ hasta que sólo quede un elemento.

```
MODULE SeleccionDirecta;
FROM InOut IMPORT WriteInt, WriteLn, WriteString;
       TYPE tipoarray=ARRAY[0..9] OF INTEGER;

       VAR i,j,k: INTEGER; n,x: INTEGER;
       VAR a: tipoarray;
BEGIN
   (* carga del arreglo de prueba en orden inverso (peor caso) *)
    n:=9;
    FOR i:=1 to 9 DO
        a[i]:=10-i
    END;

   (* Algoritmo de clasificación ------------------------------*)
    FOR i:=1 TO n-1 DO
        k:=i;
        x:=a[i]; (* Seleccion inicial *)
        FOR j:=i+1 TO n DO (* busqueda en secuencia fuente *)
            IF a[j]<x THEN
               k:=j;
               x:=a[k]; (* seleccion *)
            END
        END;
        a[k]:=a[i]; (* intercambio *)
        a[i]:=x;
    END;
   (* Fin Algoritmo de clasificación -------------------------*)

   (*Visualizacion por pantalla de pasos intermedios*)
        FOR i:=1 TO n DO
           WriteInt(a[i],4)
        END;
        WriteLn;
END SeleccionDirecta.
```

Análisis

Comparaciones

El número de comparaciones es independiente del orden inicial de llaves $n-1, n-2, \ldots, 1$, sumando todas ellas se observa que es una suma aritmética y por tanto el número de comparaciones será de orden n^2 .Así se tiene:

$$C=\frac{(n^2-n)}{2}$$

Movimientos

a) Mejor caso:

Las llaves iniciales están ordenadas $M_{min}=3(n-1)$

b) Peor caso:

Las llaves iniciales están completamente desordenadas (en orden inverso) $M_{max}=3(n-1)+\frac{n^2}{4}$

c) Caso promedio

En el caso promedio, las llaves iniciales están ordenadas aleatoriamente y habrá que utilizar argumentos probabilísticos. Dado que el procedimiento rastrea el arreglo, comparando cada elemento con el valor mínimo descubierto hasta el momento y, si es más pequeño hace una asignación. La probabilidad de que el segundo elemento sea menor que el primero es 1⁄2, la del tercero 1/3, la del cuarto 1⁄4, así sucesivamente, con lo que el número total esperado de movimiento es Hn-1, donde Hn es el enésimo número armónico

Hn= 1 + 1⁄2 + 1/3 + ... + 1/n

que se puede expresar como:

Hn= ln n + g + 1⁄2 n - 1/12n2+ ...

donde g es la constante de Euler. Con una n suficientemente grande se puede aproximar el número promedio de asignaciones como: Fi= ln i + g + l

El número promedio de movimientos en la clasificación por selección, es pues, $C=\int_1^n (\ln x)dx$, aproximando la integral se obtiene el valor aproximado $M_{prom}=n(\ln+k)$

Este resultado permite concluir que **el método de selección directa es preferible al de inserción directa** ya que su número de movimientos promedio es de orden $\ln n$. Pero si se compara con la inserción binaria cabe preguntarse cuál es más adecuado ya que éste tiene un coste de n^2 para los movimientos pero de orden $\log n$ para las comparaciones, mientras que en la selección directa es de orden n^2 para las comparaciones y de $\ln n$ para los movimientos. **La decisión en esta situación viene determinada por consideraciones de bajo nivel**. En general, la implementación de una comparación tiene un coste inferior al del 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.

Intercambio directo (burbuja)

La idea básica es comparar parejas de elementos contiguos e intercambiarlos si es necesario.

```
MODULE IntercambioDirecto;
FROM InOut IMPORT WriteInt, WriteLn, WriteString;
       TYPE tipoarray=ARRAY[0..9] OF INTEGER;

       VAR i,j: INTEGER; n,aux: INTEGER;
       VAR a: tipoarray;
BEGIN
   (* carga del arreglo de prueba en orden inverso (peor caso) *)
    n:=9;
    FOR i:=1 to 9 DO
        a[i]:=10-i
    END;

   (* Algoritmo de clasificación ------------------------------*)
    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;
   (* Fin Algoritmo de clasificación -------------------------*)

   (*Visualizacion por pantalla de pasos intermedios*)
        FOR i:=1 TO n DO
           WriteInt(a[i],4)
        END;
        WriteLn;
END IntercambioDirecto.
```

Análisis

Las operaciones más frecuentes características del algoritmo son la comparación de elementos del arreglo y los movimientos entre elementos del arreglo. También **como en la selección directa, el número de comparaciones es independiente del orden inicial de las llaves del arreglo** ya que siempre se recorre por completo la parte desordenada del arreglo para comparar las parejas de llaves adyacentes.

Comparaciones

El número de comparaciones es independiente del orden inicial de llaves $n-1, n-2, \ldots, 1$, sumando todas ellas se observa que es una suma aritmética y por tanto el número de comparaciones será de orden n^2 .Así se tiene:

$$C=\frac{(n^2-n)}{2}$$

Movimientos

a) Mejor caso:

Las llaves iniciales están ordenadas $M_{min}=0$, nunca se entra en el IF, por tanto el intercambio formado por tres movimientos, no se realiza nunca

b) Peor caso:

Las llaves iniciales están completamente desordenadas (en orden inverso) $M_{max}=3\frac{(n^2-n)}{2}$. En este caso se realiza siempre la comparación.

c) Caso promedio

En el caso promedio, las llaves iniciales están ordenadas aleatoriamente y habrá que utilizar argumentos probabilísticos. El intercambio se realizará dependiendo de que un número entero sea mayor o menor que otro, hecho de probabilidad ½ suponiendo equiprobables los enteros. Así el número promedio de movimientos es: $M_{prom}=\frac{1}{2}(M_{max})$ o lo que es lo mismo: $M_{prom}=3\frac{(n^2-n)}{4}$.

Sacudida o vibración

Es un método de clasificación por intercambio que mejora el método de la clasificación por burbuja (intercambio directo) alternando la dirección de pases consecutivos. Los pases impares se harán en sentido descendente, mientras que los pares lo harán en sentido ascendente. En cada pasada, la parte del arreglo a ordenar estará marcada por los índices L y R que indicarán respectivamente el inicio y el final de dicha parte. En cada pasada hacia el final se decrementará (R), ya que el último elemento se coloca en su sitio y en cada pasada hacia el inicio se incrementará el índice de inicio (L) por la razón, simétrica de la anterior, de que el menor de los elementos ha alcanzado su posición definitiva. El algoritmo se detiene cuando estos índices se cruzan.

```
MODULE SacudidaVibracion;
FROM InOut IMPORT WriteInt, WriteLn, WriteString;
       TYPE tipoarray=ARRAY[0..9] OF INTEGER;

       VAR j,k,L,R: INTEGER; n,aux,i: INTEGER;
       VAR a: tipoarray;
BEGIN
   (* carga del arreglo de prueba en orden inverso (peor caso) *)
    n:=9;
    FOR i:=1 to 9 DO
        a[i]:=10-i
    END;

   (* Algoritmo de clasificación ------------------------------*)
    L:=1; R:=n-1; k:=1;
    REPEAT
        FOR j:=L TO R DO          (* pase de bajada *)
            IF a[j]>a[j+1] THEN   (* comparar elementos contiguos *)
               aux:=a[j+1];       (* intercambiar *)
               a[j+1]:=a[j];
               a[j]:=aux;
               k:=j;
            END;
        END;
        R:=k-1;
        FOR j:=R TO L BY -1 DO    (* pase de subida *)
            IF a[j]>a[j+1] THEN   (* comparar elementos contiguos *)
               aux:=a[j+1];       (* intercambiar *)
               a[j+1]:=a[j];
               a[j]:=aux;
               k:=j;
            END;
        END;
        L:=k+1;
    UNTIL L>R;
   (* Fin Algoritmo de clasificación --------------------------*)

   (*Visualizacion por pantalla de pasos intermedios*)
        FOR i:=1 TO n DO
           WriteInt(a[i],4)
        END;
        WriteLn;
END SacudidaVibracion.
```

Análisis

La primera observación importante es que **las mejoras realizadas sobre la clasificación por burbuja (intercambio directo) no reducen el número de movimientos a realizar** ya que los intercambios para colocar en su lugar una llave serán los mismos. **La mejora se realiza únicamente en el número de comparaciones, eliminando las que se ejecutan de forma redundante**.

Comparaciones

El número mínimo de comparaciones será : $C_{min}=n-1$

El número máximo de comparaciones será: $C_{max}=n-k_1(\sqrt{n})$

El caso promedio de comparaciones será: $C_{prom}=\frac{1}{2}(n^2-n(k_2+\ln n))$, siendo k_1 y k_2 constantes de proporcionalidad. **Teniendo en cuenta que la implementación de los movimientos tiene un coste mayor que el de las comparaciones puede decirse que la mejora no es significativa.**

Comparación de los métodos directos de clasificación

De los análisis de los costes computacionales de realizados para cada algoritmo puede establecerse que el coste computacional de los algoritmos de clasificación directa es de orden n^2 .

Entre ellos la selección directa será la opción a elegir en general, aunque 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.