Algoritmos elementales para ordenar un vector

Pablo Santamaría

v0.1 (Junio 2007)

1. Introducción

Una de las operaciones más importantes (y demandantes) en computación es *ordenar* un cierto conjunto de datos. Para nuestros propósitos de cálculo científico, tal colección de datos es simplemente un conjunto de N valores numéricos almacenados en un *arreglo unidimensional* (*vector*) A. Dicho vector estará en *orden ascendente* si

$$i < j \qquad \text{implica que} \qquad A(i) \leq A(j)$$

para todos los elementos del vector.

En este apunte presentamos tres métodos elementales para ordenar un vector en orden ascendente, a saber, el *método de burbuja*, el *método de selección* y el *método de inserción*. Pero antes de presentarlos tenemos que enfatizar que éstos métodos *no* son los adecuados para uso general, sobre todo si el vector consta de una gran número de elementos (digamos, $N > 50$).

Para aclarar ésto tenemos que hablar de la *eficiencia* de los métodos de ordenación. La mejor manera de medir el rendimiento de un algoritmo de ordenación consiste en contar el número de comparaciones entre elementos utilizados para ordenar un vector de N elementos. Un algoritmo de ordenación será más eficiente cuanto menor sea tal número de comparaciones. En el caso de los algoritmos mencionados puede demostrarse que tal número de comparaciones es proporcional a N^2. Así, si se duplica el tamaño del vector, el número de comparaciones se cuadriplica. Por otra parte, los mejores algoritmos de ordenación tienen un número de comparaciones proporcional a $N \log N$. Claramente estos algoritmos son computacionalmente más eficientes que los algoritmos elementales presentados aquí [1].

2. Método de burbuja

El algoritmo de ordenación de burbuja ("*bubble sort*") se basa en comparar los elementos adyacentes del vector e intercambiar los mismos si están desordenados. Se comienza comparando el primer elemento con el segundo, si están desordenados se intercambian. Luego se compara el segundo con el tercero, intercambiándolos si están desordenados. Este proceso que se realiza sobre todos los elementos constituye una *pasada* sobre el vector. Al terminar esta pasada el mayor elemento se encuentra al final del vector y algunos de los elementos más pequeños se han movido hacia las primeras posiciones (es decir, los elementos más pequeños han "burbujeado" hacia arriba, mientras que los más grandes se han "hundido", de aquí el nombre del método). Ahora se vuelve a repetir el procedimiento, pero sin comparar el mayor elemento que ya se encuentra en su posición correcta. Al terminar esta segunda pasada se habrá logrado poner el segundo mayor elemento en la posición correcta. El procedimiento se vuelve a repetir hasta obtener el vector ordenado, lo que ocurrirá cuando se hayan realizado $N - 1$ pasadas.

[1] Puede verse una implementación de los algoritmos más avanzados en el capítulo 8, *Sorting*, del *Numerical Recipes in Fortran*.

```
      SUBROUTINE ORDEN(NELEM,ARREG)
*     ------------------------------------------------
*     ORDENACION POR BURBUJA ("buble sort") de un arreglo
*     unidimensional, de menor a mayor.
*
*     NELEM  = Número de elementos del arreglo
*     ARREG  = Arreglo unidimensional a ordenar
*     ------------------------------------------------
      IMPLICIT NONE
      INTEGER NELEM
      REAL ARREG(*)
*     ------------------------------------------------
      INTEGER I,J
      REAL AUX
*     ------------------------------------------------
      IF (NELEM.LT.2) RETURN
      DO I=1,NELEM-1
          DO J=1,NELEM-I
              IF (ARREG(J).GT.ARREG(J+1)) THEN
                  AUX        = ARREG(J)
                  ARREG(J)   = ARREG(J+1)
                  ARREG(J+1) = AUX
              ENDIF
          ENDDO
      ENDDO
      RETURN
      END
```

Nótese que si en una pasada no se realizan intercambios, entonces el vector se encuentra, en tal instancia, ya ordenado. Utilizando un indicador que registre si se han producido o no intercambios en la pasada el algoritmo puede ser mejorado como sigue.

```
      SUBROUTINE ORDEN(NELEM,ARREG)
*     ------------------------------------------------
*     ORDENACION POR BURBUJA ("buble sort") MEJORADO
*     de un arreglo unidimensional, de menor a mayor.
*
*     NELEM  = Número de elementos del arreglo
*     ARREG  = Arreglo unidimensional a ordenar
*     ------------------------------------------------
      IMPLICIT NONE
      INTEGER NELEM
      REAL ARREG(*)
*     ------------------------------------------------
      LOGICAL CLAVE
      INTEGER I, J
      REAL AUX
*     ------------------------------------------------
      IF (NELEM.LT.2) RETURN
      I     = 1
      CLAVE = .TRUE.
      DO WHILE(CLAVE)
          CLAVE = .FALSE.
          DO J=1,NELEM-I
```

```
            IF (ARREG(J).GT.ARREG(J+1)) THEN
                AUX        = ARREG(J)
                ARREG(J)   = ARREG(J+1)
                ARREG(J+1) = AUX
                CLAVE      = .TRUE.
            ENDIF
        ENDDO
        I = I+1
    END DO
    RETURN
    END
```

3. Método de selección

El algoritmo de ordenación por selección procede encontrando el mayor elemento del vector e intercambiando su posición con el último elemento. A continuación se repite el procedimiento sobre los elementos 1 ... N-1, y así sucesivamente.

```
      SUBROUTINE ORDEN (NELEM,ARREG)
*     ------------------------------------------------
*     ORDENACION POR SELECCION de un arreglo unidimensional,
*     de menor a mayor.
*
*     NELEM  = Número de elementos del arreglo
*     ARREG  = Arreglo unidimensional a ordenar
*     ------------------------------------------------
      IMPLICIT NONE
      INTEGER NELEM
      REAL ARREG(*)
*     ------------------------------------------------
      INTEGER I, J, INDICE
      REAL AUX
*     ------------------------------------------------
      IF (NELEM.LT.2) RETURN
      DO J= NELEM,2,-1
          INDICE = 1
          DO I =2,J
              IF (ARREG(I).GT.ARREG(INDICE)) INDICE = I
          ENDDO
          IF (INDICE.NE.J) THEN
              AUX           = ARREG(INDICE)
              ARREG(INDICE) = ARREG(J)
              ARREG(J)      = AUX
          ENDIF
      ENDDO
      RETURN
      END
```

4. Método de inserción

El algoritmo de ordenación por inserción procede sobre cada elemento insertándolo en el lugar que le corresponde a la vez que desplaza los siguientes. Este es el método usual en que un jugador

de cartas ordena las mismas.

```
      SUBROUTINE ORDEN(NELEM,ARREG)
*     ------------------------------------------------
*     ORDENACION POR INSERSION de un arreglo unidimensional,
*     de menor a mayor.
*
*     NELEM  = Número de elementos del arreglo
*     ARREG  = Arreglo unidimensional a ordenar
*     ------------------------------------------------
      IMPLICIT NONE
      INTEGER NELEM
      REAL ARREG(*)
*     ------------------------------------------------
      LOGICAL CLAVE
      INTEGER I, J, K, POS
      REAL AUX
*     ------------------------------------------------
      IF (NELEM.LT.2) RETURN
      DO I=2,NELEM
          K     = I
          AUX   = ARREG(K)
          CLAVE = .FALSE.
          DO WHILE((K.GT.1).AND.(.NOT.CLAVE))
              IF (ARREG(K-1).GT.AUX) THEN
                  ARREG(K)  = ARREG(K-1)
                  K         = K-1
              ELSE
                  CLAVE = .TRUE.
              ENDIF
          END DO
          POS        = K
          ARREG(POS) = AUX
      ENDDO
      RETURN
      END
```