Métodos Numéricos y Simulaciones en Astrofísica

Parte 3: Ordenamiento de Arrays

Algoritmos de Ordenación

- Es un algoritmo para poner los elementos de una lista en un cierto orden.
- El algoritmo debe ser eficiente.
- La cantidad de memoria utilizada es importante.
- Es importante la complejidad: O(n log n) contra $O(n^2)$.
- Recursión y adaptabilidad.

Algoritmos de Ordenación

- Burbuja.
- Inserción.
- Método de Shell.
- Heapsort.
- Quicksort.

Algoritmos de Ordenación
Método de la Burbuja

- Es un algoritmo muy simple.
- Tiene una complejidad de $O(n^2)$.
- Se basa en el intercambio de elementos.
- No es práctico ni eficiente para listas muy grandes.
- Los valores pequeños al final se mueven muy lentamente.
- Casi no se usa en trabajos serios.

Algoritmos de Ordenación
Método de la Burbuja

```
procedure burbuja-mejorado ( A : lista de objetos a ordenar )
   repeat
     cambio = false
     for i = 2 to length(A)  do:
       if A[i-1] > A[i] then
         swap( A[i-1], A[i] )
         cambio = true
       end if
     end for
   until not cambio
end procedure
```

Algoritmos de Ordenación
Método de la Burbuja

```
procedure burbuja-mejorado ( A : lista de objetos a ordenar )
  n = length(A)
   repeat
     new = 0
      for i = 2 to n  do:
         if A[i-1] > A[i] then
           swap( A[i-1], A[i] )
           new= i
         end if
     end for
     n = new
   until n = 0
end procedure
```

Algoritmos de Ordenación
Método de Inserción

- Es un algoritmo muy simple.
- Tiene una complejidad de $O(n^2)$.
- Ordena la lista de un elemento por vez.
- Usualmente no usa memoria extra.
- Es útil para menos de 20 elementos.
- No es práctico ni eficiente para listas muy grandes.

Algoritmos de Ordenación
Método de Inserción

```
procedure insercion ( A : lista de objetos a ordenar )
   for i = 2 to length(A) do:
      ele = A [ i ]
      j = i – 1
      while ( A[ j ] > ele ) and ( j > 0) then
        swap ( A[ j ] , A [ j+1 ] )
        j = j – 1
      end while
    end for
end procedure
```

Algoritmos de Ordenación
Método de Shell

- Es un algoritmo propuesto por D. Shell en 1959.
- Es una variante del método de inserción.
- Es un algoritmo de pasada multiple.
- Ordena elementos que se encuentran a cierta distancia (incrementos) usando inserción.
- Los incrementos se elijen de diferente manera. La serie original propuesta por Shell es: N/2, N/4,…, 1. Knuth (1973) propone $(3^k - 1)/2$.
- Más recientemente, M. Ciura (2001) propuso: 701, 301, 132, 57, 23, 10, 4 y 1.
- Es útil para menos de 50 elementos.

Algoritmos de Ordenación
Método de Shell

```
procedure shell ( A : lista de objetos a ordenar )
  n = length(A)
  gaps=[701, 301, 132, 57, 23, 10, 4, 1]
   for i = 1 to length(gaps) do:
      gap = gaps( i )
      k = 1
      for j = gap to n step gap do:
         B[ k ] = A[ j ]
         k = k+1
      end for
      call insercion (B)
   end for
end procedure
```

Algoritmos de Ordenación
Método Heapsort

- Es un algoritmo de comparación en dos pasos propuesto por J. Williams.
- Primero, arma una pila según un cierto criterio de selección:

Si tenemos un conjunto a_i , i=1,2,…,N se arma la pila respetando la relación

$$a_{j/2} > a_j$$

para $1 <= j/2 < j <= N$

Algoritmos de Ordenación
Método Heapsort

- Segundo, se genera una ordenación removiendo sistemáticamente el mayor elemento de la pila.
- La pila se reconstruye luego de cada paso.
- No requiere memoria extra y es relativamente fácil de programar.
- En el peor de los casos tiene una complejidad de O(n log n).

Algoritmos de Ordenación
Quicksort

- Es un algoritmo propuesto por C. Hoare.
- El algoritmo se basa en elegir un elemento pivot y separar los elementos en mayores y menores que este valor. El proceso se repite hasta ordenar.
- Cuando quedan pocos elementos es más rápido utilizar inserción directamente.
- Necesita memoria extra y la codificación es algo complicada.

Algoritmos de Ordenación
Quicksort

- Para más de 1000 elementos, Quicksort es 1.5 – 2 veces más rápido que cualquier otro algoritmo.
- En el mejor de los casos tiene una complejidad de O(n log n) pero en el peor es $O(n^2)$.
- El peor caso es cuando el array ya esta ordenado.