Sorting / Ordenación

Insertion Sort

→ En el i-ésimo recorrido se inserta el i-ésimo elemento en el lugar correcto entre los (i-1) elementos anteriores ordenados.

Util para arreglos pequeños o casi ordenados.

Pseudocódigo:

```
para i desde 2 hasta N hacer
    Aux ← V[i]
    j ← i - 1
    mientras j > 0 y Aux.clave < V[j].clave hacer
        V[j+1] ← V[j]
        j ← j - 1
    fin mientras
    V[j+1] ← Aux
```

Selection Sort

Selecciona el menor elemento del subconjunto no ordenado y lo coloca en su posición correcta.

Pseudocódigo:

```
para i desde 1 hasta N - 1 hacer
    IndiceMenor ← i
    ClaveMenor ← V[i].clave
    para j desde i+1 hasta N hacer
        si V[j].clave < ClaveMenor entonces
```

```
        IndiceMenor ← j
        ClaveMenor ← V[j].clave
    intercambiar V[i], V[IndiceMenor]
```

Shellsort

Divide el array en subgrupos con un 'gap', realiza inserción en cada grupo, reduce el gap y repite.

Pseudocódigo:

```
 elegir secuencia de gaps (n/2, n/4, ..., 1)
para cada gap hacer
    para i desde gap hasta N hacer
        insertar elemento V[i] en su posición dentro del subgrupo separado por gap
```

Quicksort

Algoritmo divide y vencerás. Usa un pivote para particionar y ordena recursivamente.

Pseudocódigo:

```
 quicksort(i, j)
    si i < j entonces
        pivote ← V[IndicePivote]
        k ← particion(i, j, pivote)
        quicksort(i, k - 1)
        quicksort(k, j)
```

Mergesort

Divide el arreglo en mitades, ordena recursivamente y fusiona.

Pseudocódigo:

```
procedimiento mergesort(A)
    si largo(A) <= 1 entonces retornar
    dividir A en L y R
    mergesort(L)
    mergesort(R)
    fusionar(L, R) → A
```

Heapsort

Utiliza un heap también llamado montículo para ordenar los elementos.

Pseudocódigo:

```
para x en E hacer INSERTA(x, S)
mientras no VACIA(S)
    y ← MIN(S)
    procesar(y)
   SUPRIME(y, S)
```

Construcción del heap:

```
para i desde ⌊n/2⌋ hasta 1 hacer
   DesplazaElemento(i, n)
```

Bucket/Bin Sort

Distribuye elementos en cubos según su clave, luego ordena y concatena.

Pseudocódigo:

```
crear m cubos (urnas vacías)
```

```
para cada elemento e en entrada
   insertar e en cubo según clave(e)
para cada cubo ordenar si se desea
concatenar cubos en orden
```

Radix Sort

Ordena campos desde el menos al más significativo, usando ordenación estable.

Pseudocódigo:

```
para i desde k hasta 1 hacer
  vaciar Bi[v] para cada valor v
   para cada registro R en A hacer
      insertar R en Bi[valor(R.fi)]
   A ← concatenación ordenada de Bi
```

Bubble Sort

Intercambia pares adyacentes si están en orden incorrecto.

Pseudocódigo:

```
para i desde 1 hasta N - 1 hacer
   intercambiado ← falso
   para j desde 1 hasta N - i hacer
      si A[j] > A[j+1] entonces
         intercambiar A[j], A[j+1]
         intercambiado ← verdadero
   si no intercambiado entonces salir
```