**ORDENAMIENTO**

**Es la operación de arreglar los registros de una tabla en algún orden secuencial de acuerdo a  un criterio de ordenamiento. El ordenamiento se efectúa con base en el valor de algún campo  en  un  registro. El  propósito principal  de un ordenamiento es el de facilitar las búsquedas de los miembros del conjunto ordenado.**

**El  ordenar  un  grupo  de  datos  significa mover los  datos  o  sus  referencias  para  que queden  en  una  secuencia tal  que  represente  un  orden,  el  cual  puede  ser  numérico, alfabético o incluso alfanumérico, ascendente o descendente.**

**Algoritmos de ordenación**

**Los algoritmos básicos de ordenación más simples y clásicos son:**

* **Ordenación por selección.**
* **Ordenación por inserción.**
* **Ordenación por burbuja.**

**Los métodos más recomendados son el de selección y el de inserción, aunque el de burbuja es el más sencillo aunque el más ineficiente.**

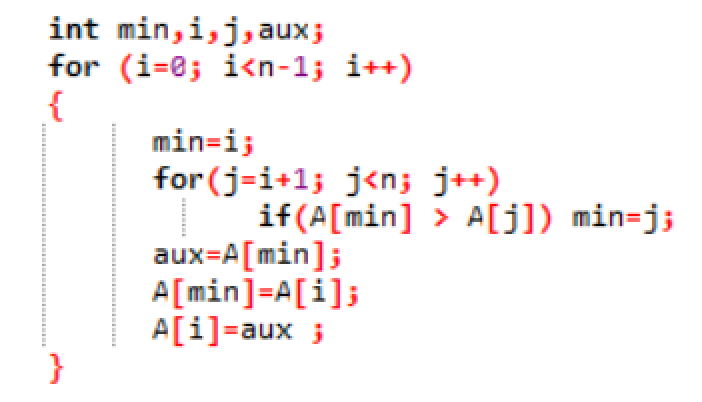
**Algoritmo de selección**

**Uno de los algoritmos de ordenación más sencillos funciona de la siguiente forma: primero se busca el elemento más pequeño del array y se intercambia con el que está en la primera posición; después se busca el segundo elemento más pequeño y se intercambia con el que está en la segunda posición, continuándose de esta forma hasta que todo el array esté ordenado**

**Los pasos del algoritmo son:**

* **Seleccionar el elemento más pequeño de la lista A. Intercambiarlo con el primer elemento A [0]. Ahora la entrada más pequeña está en la primera posición del vector.**
* **Considerar las posiciones de la lista A[1], A[2], A[3]…, seleccionar el elemento más pequeño e intercambiarlo con A[1]. Ahora las dos primeras entradas de A están en orden.**
* **Continuar este proceso encontrando o seleccionando el elemento más pequeño de los restantes elementos de la lista, intercambiándolos adecuadamente.**

**El algoritmo es:**

****

**ANÁLISIS DEL ALGORITMO.**

**• Requerimientos   de Memoria:  Al   igual   que   el   ordenamiento   burbuja, este algoritmo sólo necesita una variable adicional para realizar los intercambios.**

**• Tiempo de Ejecución: El ciclo externo se ejecuta n  veces para una lista de n elementos.   Cada   búsqueda   requiere   comparar   todos   los   elementos   no clasificados.**

**Ventajas:**

**• Fácil implementación.**

**• No requiere memoria adicional.**

**• Rendimiento constante: poca diferencia entre el peor y el mejor caso.**

**Desventajas:**

**• Lento.**

**• Realiza numerosas comparaciones.**

**Por ejemplo tenemos el vector A={51, 21, 39, 80, 36}**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **51** | **21** | **39** | **80** | **36** |   **Pasada 0 A[0] A[1] A[2] A[3] A[4]** | | | | | | | **Pasada 0: Seleccionar 21 Intercambiar 21 y A[0]** |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **21** | **51** | **39** | **80** | **36** |   **Pasada 1** | | | | | | | **Pasada 1: Seleccionar 36 Intercambiar 36 y A[1]** |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **21** | **36** | **39** | **80** | **51** |   **Pasada 2** | | | | | | | **Pasada 2: Seleccionar 39 Intercambiar 39 y A[2]** |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **21** | **36** | **39** | **80** | **51** | | | | | | | | **Pasada 3: Seleccionar 51 Intercambiar 51 y A[3]** |
| **Pasada 3** | | | | | | |  |
| **21** | **36** | **39** | **51** | **80** |
| **LISTA ORDENADA** | | | | | | |  |

**ORDENACIÓN POR INSERCIÓN LINEAL O DIRECTA**

**El método de ordenación por inserción se similar al proceso típico de ordenar tarjetas de nombres (cartas de una baraja) por orden alfabético, que consiste en insertar un nombre en su posición correcta dentro de una lista o archivo que ya está ordenado.**

**DESCRIPCIÓN.**

**El   algoritmo   de   ordenación   por   el   método   de   inserción   directa   es   un   algoritmo relativamente   sencillo   y   se   comporta   razonablemente   bien   en   gran   cantidad   de situaciones.**

**Completa   la   tripleta   de   los   algoritmos   de   ordenación   más   básicos   y   de   orden   de complejidad cuadrático, junto con SelectionSort y BubbleSort.**

**Se basa en intentar construir una lista ordenada en el interior del array a ordenar.**

**De estos tres algoritmos es el que mejor resultado da a efectos prácticos. Realiza una cantidad de comparaciones bastante equilibrada con respecto a los intercambios, y tiene un par de características que lo hacen aventajar a los otros dos en la mayor parte de las situaciones.**

**Este algoritmo se basa en hacer comparaciones, así que para que realice su trabajo de ordenación son imprescindibles dos cosas: un array o estructura similar de elementos comparables y un criterio claro de comparación, tal que dados dos elementos nos diga si están en orden o no.**

**ANÁLISIS DEL ALGORITMO.**

**• Estabilidad: Este algoritmo nunca intercambia registros con claves iguales. Por lo tanto es estable.**

**• Requerimientos   de   Memoria:   Una   variable   adicional   para   realizar   los intercambios.**

**• Tiempo de Ejecución: Para una lista de n elementos el ciclo externo se ejecuta n­1  veces. El ciclo  interno  se  ejecuta como máximo  una  vez  en la  primera iteración, 2 veces en la segunda, 3 veces en la tercera, etc.**

**Ventajas:**

**• Fácil implementación.**

**• Requerimientos mínimos de memoria.**

**Desventajas:**

**• Lento.**

**• Realiza numerosas comparaciones.**

**Este también es un algoritmo lento, pero puede ser de utilidad para listas que están ordenadas o semi­ordenadas, porque en ese caso realiza muy pocos desplazamientos.**

**Por ejemplo para la siguiente lista de enteros A= {50,20,40,80,30}**

|  |
| --- |
| **50 Comienzo con 50** |
| **Procesar 20 20 50 .Se inserta 20 en la posición 0;**  **.50 se mueve a posición 1** |
| **Procesar 40 20 40 50 .Se inserta 40 en la posición1;**  **.Se mueve 50 a posición 2** |
| **Procesar 80 20 40 50 80 .El elemento 80 está bien ordenado** |
| **Procesar 30 20 30 40 50 80 .Se inserta 30 en posición 1**  **.Se desplaza a la derecha la sublista derecha** |

**La función Ordenar por inserción lineal tiene dos parámetros, el array A y el tamaño de la lista n. el algoritmo correspondiente contempla los siguientes pasos:**

1. **El primer elemento A[0] se considera ordenado; es decir, la lista inicial consta de un elemento.**
2. **Se inserta A[1] en la posición correcta; delante o detrás de A[0], dependiendo de que sea mayor o menor. Es decir, se explora la lista desde A[i] hasta A[n] buscando la posición correcta de destino; esto es, la posición a insertar dentro de una lista ordenada.**
3. **Por cada bucle o iteración se mueve hacia abajo (a la derecha en la lista) una posición todos los elementos mayores que la posición a insertar, para dejar vacía esa posición.**
4. **Insertar el elemento a la posición correcta.**

**ORDENACIÓN POR INTERCAMBIO**

**Uno de los algoritmos de ordenación más sencillo es del intercambio, que ordena los elementos de una lista en orden ascendente. El algoritmo se basa en la lectura sucesiva de la lista a ordenar, comparando el elemento inferior de la lista con los restantes y efectuando intercambio de posiciones cuando el orden resultante de la comparación no sea el correcto.**

**Por ejemplo tenemos la lista A={8,4,6,2}**

**El algoritmo realiza n-1 pasadas, realizando las siguientes operaciones:**

**Pasada 0**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A[0] A[1] A[2] A[3]**   |  |  |  |  | | --- | --- | --- | --- | | **8** | **4** | **6** | **2** | | **Se realiza intercambio** | |  |  |  |  | | --- | --- | --- | --- | | **4** | **8** | **6** | **2** | |
| |  |  |  |  | | --- | --- | --- | --- | | **4** | **8** | **6** | **2** | | **No se realiza intercambio** | |  |  |  |  | | --- | --- | --- | --- | | **4** | **8** | **6** | **2** | |
| |  |  |  |  | | --- | --- | --- | --- | | **4** | **8** | **6** | **2** | | **Se realiza intercambio** | |  |  |  |  | | --- | --- | --- | --- | | **2** | **8** | **6** | **4** | |

**El elemento de índice 0 (A[0]) se compara con cada elemento posterior de la lista de índices 1,2 y 3. En cada comparación se comprueba si el elemento siguiente es más pequeño que el elemento de índice 0, en ese caso se intercambian. Después de terminar todas las comparaciones el elemento más pequeño se localiza en el índice 0.**

**Pasada 1**

**El elemento más pequeño ya está localizado en el índice 0, y se considera la sublista restante 8, 6, 4. El algoritmo continúa comparando el elemento, índice 1 con los elementos posteriores de índices 2 y 3. Por cada comparación, si el elemento mayor está en el índice 1 se intercambian los elementos. Después de hacer todas las comparaciones, el segundo elemento más pequeño de la lista se almacena en el índice 1.**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | **2** | **8** | **6** | **4** | | **Intercambio** | |  |  |  |  | | --- | --- | --- | --- | | **2** | **6** | **8** | **4** | |
| |  |  |  |  | | --- | --- | --- | --- | | **2** | **6** | **8** | **4** | | **Intercambio** | |  |  |  |  | | --- | --- | --- | --- | | **2** | **4** | **8** | **6** | |

**Pasada 2**

**La sublista a considerar ahora es 8, 6 qye 2, 4 está ordenada. Una comparación única se produce entre los dos elementos de la sublista.**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | **2** | **4** | **8** | **6** | | **Intercambio** | |  |  |  |  | | --- | --- | --- | --- | | **2** | **4** | **6** | **8** |   **Lista resultante** |