Universidad Autónoma del Estado de Hidalgo

Instituto de Ciencias Básicas e Ingeniería

Área Académica de Computación y Electrónica

Licenciatura en Sistemas Computacionales

Estructura de Datos II

Docente: M. en C. Fabiola Martínez Juárez

Tema:
Métodos de Ordenación Básicos

Resumen:

Esta presentación expone los métodos de ordenación básicos, como son: ordenación por selección, ordenación por inserción y ordenación por burbuja. De cada uno de ellos, se presenta su enfoque, un ejemplo y las generalidades de su algoritmo.

Palabras clave: *Ordenación*, *Algoritmos*, *Selección*, *Inserción*, *Burbuja*.

Tema:
Métodos de Ordenación Básicos

Abstract:

This presentation outlines the basic sorting methods: selection sort, insertion sort and bubble sort. The approach, an example and an overview of the algorithm is presented for each one of them.

Keywords: *Sorting*, *Algorithms*, *Selection sort*, *Insertion sort*, *Bubble sort*.

Métodos de Ordenación Básicos

La ordenación es el proceso de reagrupar o reorganizar un conjunto de datos en orden creciente (o decreciente). Es un proceso que se practica con mucha frecuencia, y si bien muchos lenguajes proveen herramientas que ejecutan esta tarea, es muy importante el estudio de los métodos de ordenación básicos.

Dichos métodos tienen la característica de que su implementación es relativamente sencilla y son fáciles de comprender, aunque son ineficientes cuando el número de elementos a ordenar es grande [1],[2],[3],[4].

Los algoritmos son:

- *Ordenación por selección*
- *Ordenación por inserción*
- *Ordenación por burbuja*

Ordenación por Selección

Se basa en seleccionar la posición del elemento más pequeño de la lista y su colocación en la posición que le corresponde.

Como primer paso se encuentra el elemento menor de la lista y se intercambia con el elemento de subíndice **0**. Después se busca el elemento menor en la sublista formada desde el elemento en el subíndice **1** … hasta el elemento de subíndice **n-1** y se intercambia con el elemento de subíndice **1**. El proceso continúa sucesivamente durante **n-1** pasadas [1],[2],[3],[4].

n: *número de elementos en la lista.*

Ordenación por Selección

Ejemplo: Ordenar ascendentemente la lista 11, 9, 17, 5, 14 por el método de selección.

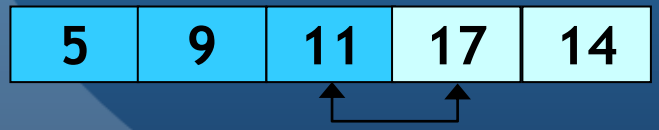
Iteración 1

- 5 es el menor y se sitúa en la primera posición.

Iteración 2

- 9 es el menor y se sitúa en la segunda posición.

Iteración 3



- 11 es el menor y se sitúa en la tercera posición.

Iteración 4

- 14 es el menor y se sitúa en la cuarta posición.

(1)

Ordenación por Selección

Generalidades del algoritmo

- Los elementos de la lista se revisan desde el subíndice **`i=0`** hasta **`n-2`**.
- Identificar la posición del elemento más pequeño de la sublista formada por los elementos desde el subíndice **`i`** hasta el subíndice **`n-1`**.
- Intercambiarlo por el primer elemento de la sublista.

***`n`**: número de elementos en la lista.*

Ordenación por Inserción

Este método es similar al proceso típico de ordenar tarjetas de nombres (p.e. cartas de una baraja) por orden alfabético, que consiste en insertar un nombre en su posición correcta dentro de una lista que ya está ordenada.

Como primer paso, el elemento `a[0]` se considera ordenado, es decir la lista inicial consta de un elemento. Después se inserta `a[1]` en la posición correcta; delante o detrás de `a[0]`, dependiendo de si es menor o mayor. Se vuelve a realizar el proceso hasta `a[n-1]` [1],[2],[3],[4].

`n`: número de elementos en la lista.

Ordenación por Inserción

Ejemplo: Ordenar ascendentemente la lista 50, 20, 40, 80, 30 por el método de inserción.

	a[0]	a[1]	a[2]	a[3]	a[4]	
	50					
Iteración 1: ***procesando 20***	20	50				• 50 se mueve a la posición 1. • Se inserta 20 en la posición 0.
Iteración 2: ***procesando 40***	20	40	50			• Se mueve 50 a posición 2. • Se inserta 40 en la posición 1.
Iteración 3: ***procesando 80***	20	40	50	80		• El elemento 80 está bien ordenado.
Iteración 4: ***procesando 30***	20	30	40	50	80	• Se mueve 80, 50 y 40. • Se inserta 30 en posición 1.

Ordenación por Inserción

Generalidades del algoritmo

- El elemento con subíndice `0` se considera ordenado.
- Se revisan los elementos desde subíndice `1` hasta `n-1`.
- Para cada elemento con subíndice `i`, la sublista `0` hasta `i-1` está ordenada. Se realiza una búsqueda lineal de su lugar correcto y se mueven los datos necesarios para colocar al elemento en su posición.

`n`: número de elementos en la lista.

Ordenación por Burbuja

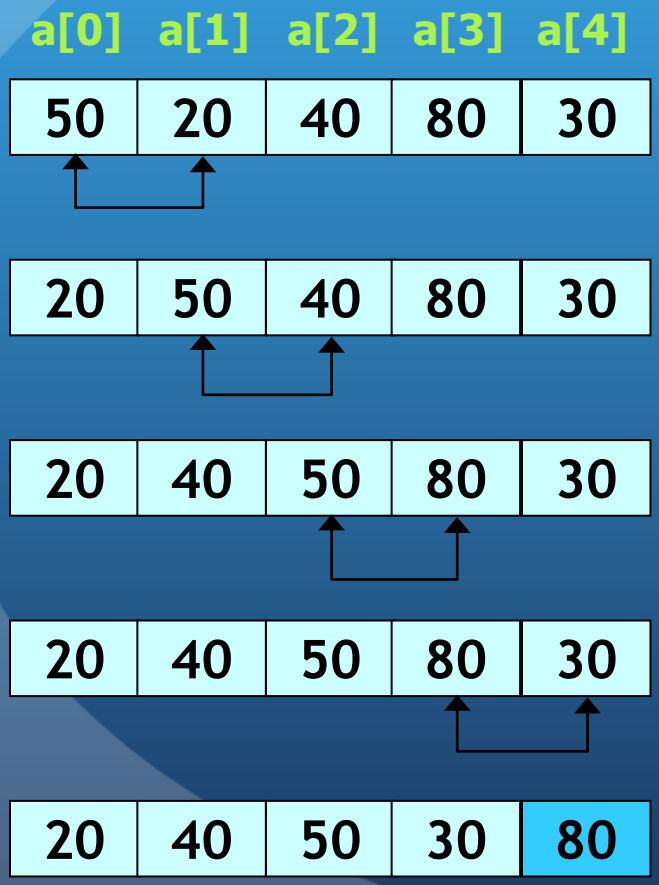
Este método utiliza una técnica donde los valores más pequeños suben a la parte superior del arreglo (burbuja), mientras que los valores mayores se hunden en la parte inferior del arreglo.

La técnica consiste en hacer varias revisiones al arreglo. En cada iteración se comparan parejas sucesivas de elementos. Si una pareja está en orden creciente (o los valores son idénticos) se dejan los valores como están. Si una pareja está en orden decreciente, sus valores se intercambian en el arreglo [1],[3],[4].

Ordenación por Burbuja

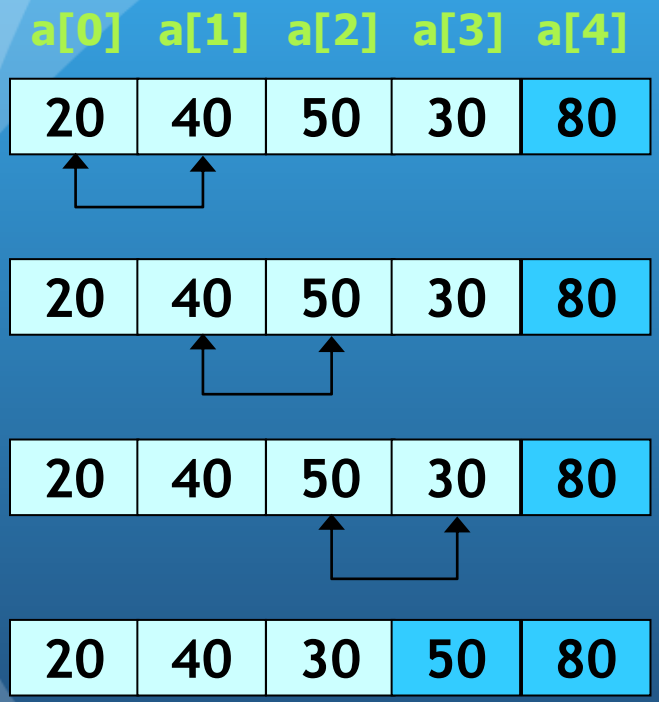
Ejemplo: Ordenar ascendentemente la lista 50, 20, 40, 80, 30 por el método de burbuja.

Iteración 1



Ordenación por Burbuja

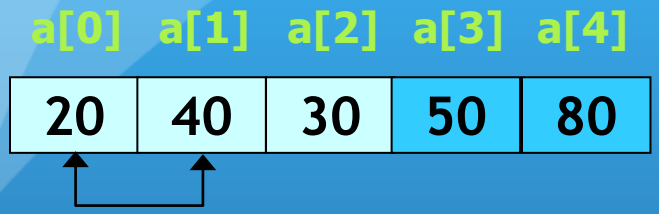
Iteración 2



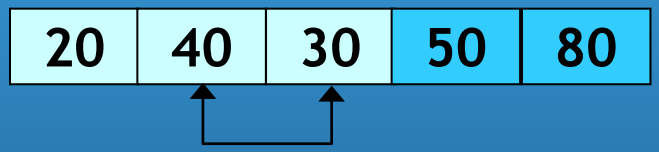
- 20 y 40 ordenados.
- 40 y 50 ordenados.
- Intercambio 50 y 30.
- 50 y 80 ordenados.

Ordenación por Burbuja

Iteración 3



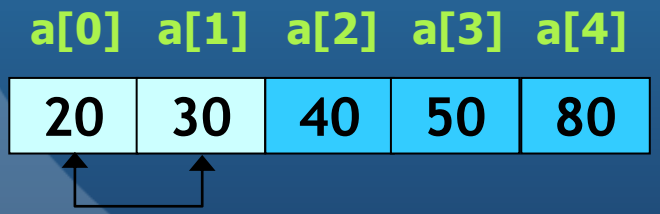
- 20 y 40 ordenados.



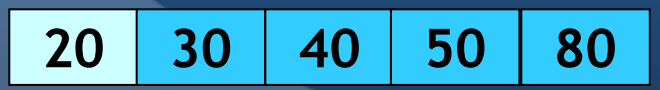
- Intercambio 40 y 30.

- 40, 50 y 80 ordenados

Iteración 4



- 20 y 30 ordenados.



- Lista ordenada.

Ordenación por Burbuja

Generalidades del algoritmo

- El la iteración `1` se comparan elementos adyacentes (`a[0]`, `a[1]`), (`a[1]`, `a[2]`), … (`a[n-2]`, `a[n-1]`). Se realizan `n-1` comparaciones.
- Por cada pareja (`a[j]`, `a[j+1]`), se intercambian los valores si `a[j]` > `a[j+1]`.
- Al final de la iteración `1`, el elemento mayor de la lista está situado en `a[n-1]`.

`n`: número de elementos en la lista.

Ordenación por Burbuja

Generalidades del algoritmo

- En la iteración **`2`** se realizan las mismas comparaciones e intercambios, terminando con el elemento de segundo mayor valor en **`a[n-2]`**.
- El proceso termina con la iteración **`n-1`**, en la que el elemento más pequeño se almacena en **`a[0]`**.

`n`: *número de elementos en la lista.*

Fuente Imágenes:

(1) Joyanes, L. & Zahonero, I. (2010) *Programación en C, C++, JAVA y UML.* México: McGrawHill.

Referencias:

[1] Cairo, O. & Guardati, S. (2006) *Estructuras de datos*. 3era ed. México: McGrawHill.

[2] Guardati, S. (2007) *Estructura de datos orientada a objetos. Algoritmos con C++.* México: Pearson Educación.

[3] Joyanes, L. & Zahonero, I. (2010) *Programación en C, C++, JAVA y UML.* México: McGrawHill.

[4] Koffman, E. B. & Wolfgang, P. A. T. (2008) *Estructuras de datos con C++. Objetos, abstracciones y diseño.* México: McGrawHill.