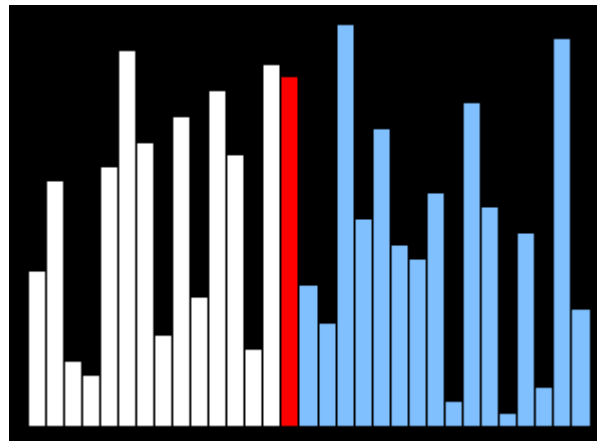




# Algoritmos y Estructuras de Datos I

Facultad de Ciencias Exactas y Tecnología  
Universidad Nacional de Tucumán  
2023

# Ordenación(1)



# Unidad IV – Ordenamiento

## **Contenidos:**

El problema de clasificación. Tiempo de ejecución. Memoria necesaria. Estabilidad. Sensibilidad.

Métodos simples: Ordenación por selección, por inserción directa, por intercambio directo. Complejidad de los algoritmos simples. Limitaciones.

Métodos mejorados: Ordenación por el método de incrementos decrecientes, Ordenación por el método rápido, Ordenación por mezcla. El tipo abstracto de datos Cola de Prioridad. Montículo. Ordenación por el método del montículo. Método de ordenación por residuos. Análisis de la complejidad de cada método. Comparación de los distintos métodos.

Métodos lineales de clasificación para claves particulares.

Ordenación externa. Métodos de mezcla.

# Historia

Los orígenes de las técnicas actuales de ordenación datan del siglo XIX. Herman Hollerith en 1880 diseñó una maquina tabuladora eléctrica para procesar un censo de todos los ciudadanos norteamericanos. Podía procesar unas 40 tarjetas por minuto. Para el censo de 1900, Hollerith patentó un modelo que permitía la alimentación automática. Estas maquinas son la bases de algunos algoritmos de ordenación que se usan hoy en día.

La idea de mezcla apareció hacia 1938, con una maquina que podía mezclar dos lotes de tarjetas ordenadas.

John von Newman preparó programas para ordenación interna en 1945, para la computadora EDVAC.

Recién en 1956 se publicaron los primeros artículos. Varios métodos de ordenación se han descubierto desde entonces, entre ellos Mezcla por inserción (1959), Shell (1959), Inserción en arboles (1960), Quicksort (1962), Heapsort (1964).

# Historia

Formato de tarjeta de IBM, diseñado en 1928, tenía perforaciones rectangulares, 80 columnas con 12 lugares de perforación cada una, y un caracter para cada columna.

[illegible]

# Problema de clasificación

Dada una lista de  $n$  elementos:  $x_1, x_2, x_3, \dots, x_n$

Cada elemento  $x_i$  tiene 2 partes:

clave <sub>i</sub>	información asociada a la clave <sub>i</sub>
--------------------	--

**Ordenarlos** es generar una permutación  $P$  de los elementos: que ubique a los mismos según los valores de sus claves de manera:

**Ascendente:**  $x_{p(1)} \leq x_{p(2)} \leq x_{p(3)} \leq \dots \leq x_{p(n)}$

o **descendente:**  $x_{p(1)} \geq x_{p(2)} \geq x_{p(3)} \geq \dots \geq x_{p(n)}$

# Clave de clasificación

Para poder clasificar según la **clave de clasificación** se supone que existe una **relación de orden**  $<$  definida en el tipo de las claves, de modo que valen las siguientes propiedades:

**Tricotomía:** dadas dos claves cualquiera  $c_1$  y  $c_2$  ,  
exactamente una de las tres posibilidades es verdadera:  $\left\{ \begin{array}{l} c_1 = c_2 , \\ c_1 < c_2 , \\ c_2 < c_1 \end{array} \right.$

**Transitiva:**

si  $c_1 < c_2$  y si  $c_2 < c_3$  entonces  $c_1 < c_3$

El problema planteado es ordenar un conjunto de elementos según una relación de orden definida sobre el tipo de sus claves.

# Métodos de Ordenación

Los algoritmos de ordenación pueden ser de dos tipos:

- **Clasificación interna:** es aquella que tiene lugar en la memoria principal, también se denomina ordenación de vectores o listas, que son las estructuras de datos con que se representan los datos.
- **Clasificación externa:** opera con elementos almacenados en dispositivos externos, se denomina también ordenación de archivos. Se deben usar cuando la cantidad de datos es grande y no se puede clasificar internamente porque no se pueden almacenar en memoria principal.



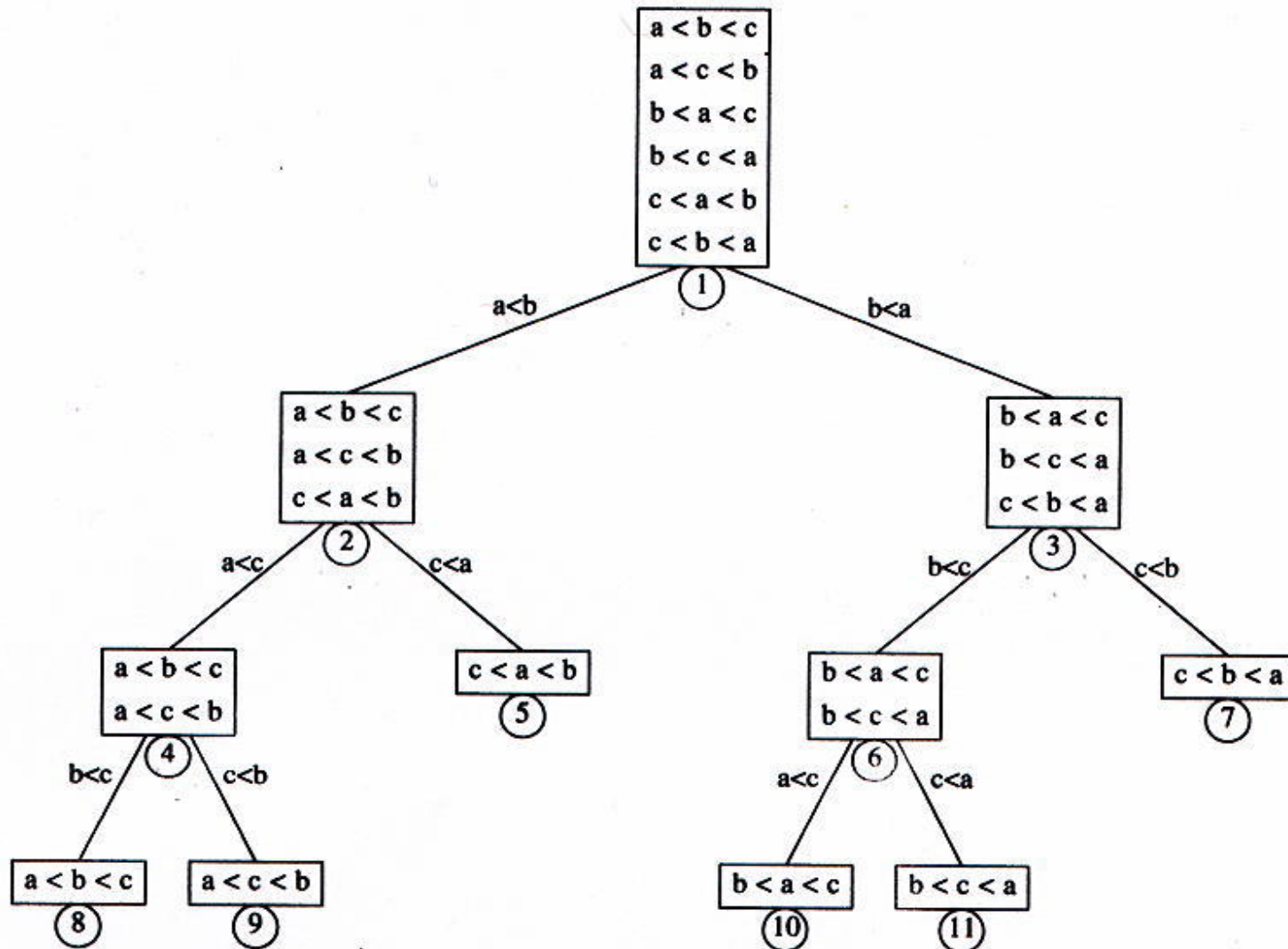
# Métodos de Ordenación Interna

El principal parámetro de rendimiento que interesa es el tiempo de ejecución de los algoritmos.

## TIEMPO DE EJECUCION

- Los métodos más simples  $\in O(n^2)$ , para ordenar  $n$  elementos.
- Los métodos más avanzados  $\in O(n \log n)$ .
- Se puede demostrar que ningún método de ordenación puede usar menos de  $(n \log n)$  comparaciones de las claves .
- Existen algunos métodos que usan propiedades especiales de las claves que pueden lograr tiempo de ejecución proporcional a  $n$ .

# Ejemplo: Ordenar (a,b,c) donde $a \neq b \neq c$



# Métodos de Ordenación Interna

El segundo factor importante a considerar es:

## CANTIDAD DE MEMORIA

Básicamente se distinguen:

- Métodos que ordenan *in situ* un arreglo y no utilizan memoria extra, salvo unas cuantas variables auxiliares.
- Métodos que ordenan *in situ* un arreglo y usan como memoria extra una o mas pilas de tamaño  $n$ , como es el caso de los algoritmos recursivos.
- Métodos que necesitan memoria extra como para almacenar una copia del arreglo que se ordena.
- Métodos que utilizan una representación por lista enlazada y necesitan por tanto  $n$  palabras de memoria para los punteros de la lista.

# Métodos de Ordenación Interna

Otro factor a considerar es:

## ESTABILIDAD

Se dice que un método de ordenación es *estable* si, cuando encuentra dos elementos que tienen la misma clave, conserva su orden relativo en el proceso de ordenación.

Por ejemplo: si se toma una lista con los estudiantes de un curso ordenados alfabéticamente y se ordenan por notas, un método estable dará una lista en la que los estudiantes que tienen las mismas notas permanecen ordenados alfabéticamente; en cambio, un método no estable es probable que dé una lista (ordenada por nota) sin que quede ningún rastro del orden alfabético original.

# Métodos de Ordenación Interna

Otro factor a considerar es:

## SENSIBILIDAD

Un método de ordenación es *sensible* respecto al orden inicial de los datos si disminuye el número de operaciones en caso de que el arreglo esté ordenado inicialmente.

Las operaciones que realizan los métodos de ordenación son fundamentalmente:

- **Comparaciones**
- **Movimientos**

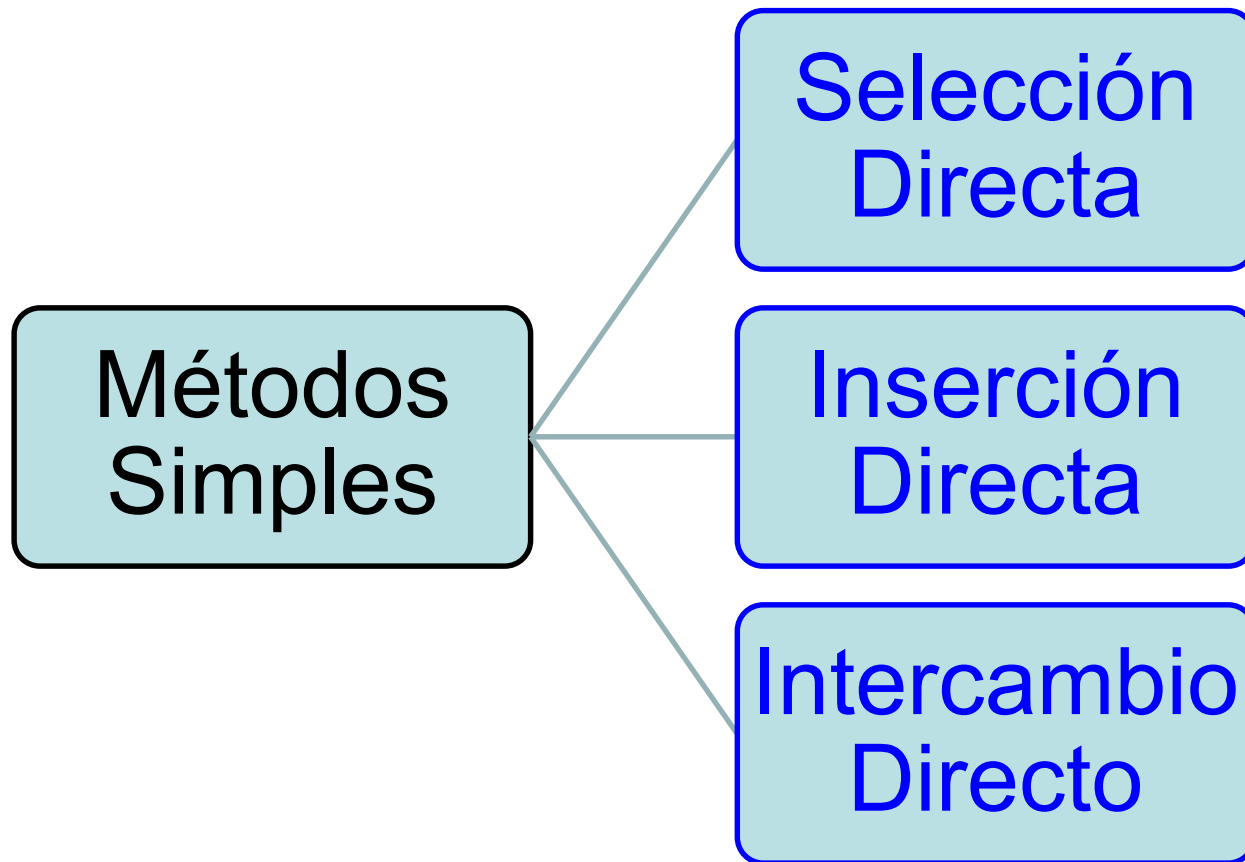
# Métodos de Ordenación

El *algoritmo ideal* para un método de ordenación de  $n$  claves tendría que tener las siguientes propiedades:

- Requerir solo  $O(1)$  almacenamiento extra para ordenar.
- Las Comparaciones de claves  $\in O(n \cdot \log(n))$ , en el peor caso.
- Los Intercambios  $\in O(n)$ , en el peor caso.
- Ser Estable: las claves iguales no intercambian su posición relativa.
- Ser Muy Sensible: disminuir el tiempo de ejecución hasta  $O(n)$  cuando los datos están casi ordenados.
- Disminuir el tiempo de ejecución hasta  $O(n)$  cuando los datos se distribuyen en pocos grupos de claves iguales.

***Ningún algoritmo cumple con todas estas propiedades a la vez.***

# Métodos de Ordenación Interna



# Método de SELECCIÓN

La idea básica es hacer “*pasadas*” en el vector, en cada pasada *seleccionar un elemento y posicionarlo* adecuadamente.

Así, para ordenar en forma ascendente, primero se busca el elemento más pequeño del arreglo y se intercambia con el que está en la primera posición, después se busca el segundo menor y se lo intercambia con el que está en la segunda posición, continuando de esta forma hasta que esté todo el arreglo ordenado.

El algoritmo se llama de ordenación por selección porque funciona seleccionando repetidamente el elemento más pequeño de los que quedan por ordenar.



# Método de SELECCIÓN

Así se construye una iteración de modo que:

**Antes del paso k:**

*ordenados* *desordenados*  
 $a_1 \leq a_2 \leq a_3 \leq \dots \leq a_{k-1}$  ,  $a_k, a_{k+1}, \dots, a_n$

**Paso k:** *seleccionar* el mínimo  $(a_k, \dots, a_n)$  e intercambiar con  $a_k$ .

**Después del paso k:**

*ordenados* *desordenados*  
 $a_1 \leq a_2 \leq a_3 \leq \dots \leq a_{k-1} \leq a_k$  ,  $a_{k+1}, \dots, a_n$

El algoritmo en su versión mas simple se puede formular como sigue<sub>17</sub>

# Algoritmo de SELECCIÓN

**Algoritmo Selección (A,n)**

**Entrada:** A: arreglo de (1.. MAX) elementos de tipoitem

n: nro. entero de datos a ordenar

**Salida:** A arreglo ordenado.

Auxiliar: i,j,kmenor :entero; menor: tipoitem

PARA i = 1 , n-1 HACER

    menor  $\leftarrow$  A(i)

    kmenor  $\leftarrow$  i

    PARA j = i+1 , n HACER

        SI A(j) < menor ENTONCES

            kmenor  $\leftarrow$  j

            menor  $\leftarrow$  A(j)

    A(kmenor)  $\leftarrow$  A(i)

    A(i)  $\leftarrow$  menor

FIN

# Método de SELECCIÓN

Ej. Ordenar el arreglo de 15 letras: ASORTINGEXAMPLE

<b>A</b>	S	O	R	T	I	N	G	E	X	A	M	P	L	E
A	S	O	R	T	I	N	G	E	X	<b>A</b>	M	P	L	E
A	A	O	R	T	I	N	G	<b>E</b>	X	S	M	P	L	E
A	A	E	R	T	I	N	G	O	X	S	M	P	L	<b>E</b>
A	A	E	E	T	I	N	<b>G</b>	O	X	S	M	P	L	R
A	A	E	E	G	<b>I</b>	N	T	O	X	S	M	P	L	R
A	A	E	E	G	I	N	T	O	X	S	M	P	<b>L</b>	R
A	A	E	E	G	I	L	T	O	X	S	<b>M</b>	P	N	R
A	A	E	E	G	I	L	M	O	X	S	T	P	<b>N</b>	R
A	A	E	E	G	I	L	M	N	X	S	T	P	<b>O</b>	R
A	A	E	E	G	I	L	M	N	O	S	T	<b>P</b>	X	R
A	A	E	E	G	I	L	M	N	O	P	T	S	X	<b>R</b>
A	A	E	E	G	I	L	M	N	O	P	R	<b>S</b>	X	T
A	A	E	E	G	I	L	M	N	O	P	R	S	X	<b>T</b>
A	A	E	E	G	I	L	M	N	O	P	R	S	T	X
A	A	E	E	G	I	L	M	N	O	P	R	S	T	X

i=1	kmenor=1	menor=A
i=2	kmenor=11	menor=A
i=3	kmenor=9	menor=E
i=4	kmenor=15	menor=E
i=5	kmenor=8	menor=G
i=6	kmenor=6	menor=I
i=7	kmenor=14	menor=L
i=8	kmenor=12	menor=M
i=9	kmenor=14	menor=N
i=10	kmenor=14	menor=O
i=11	kmenor=13	menor=P
i=12	kmenor=15	menor=R
i=13	kmenor=13	menor=S
i=14	kmenor=15	menor=T

Ordenados



# Propiedades Algoritmo de SELECCIÓN

**Tiempo de Ejecución:**  $T(n) \in O(n^2)$

**Memoria:** almacenamiento  $\in O(1)$  de espacio extra.

**Sensibilidad:** poca, solo reduce movimientos.

**Estabilidad:** NO es estable.

**Otras:** por sus características es un método que solo debe usarse para pocos datos. Sin embargo, tiene la propiedad de minimizar el número de movimientos, cuando el arreglo está ordenado se reducen a  $O(n)$ . Por lo tanto se puede usar cuando el intercambio de los elementos lleve mucho tiempo porque se trate de objetos con mucha información asociada a la clave.

# Propiedades


## Algoritmo de SELECCIÓN

**Ejemplo** para mostrar que NO es estable

$n=4$ , 3 iteraciones

Inicial  $A = [9_1, 9_2, 4, 1]$

$i=1$   $A = [9_1, 9_2, 4, \textcolor{red}{1}] \rightarrow A = [\textcolor{blue}{1}, 9_2, 4, 9_1]$



$i=2$   $A = [\textcolor{blue}{1}, 9_2, \textcolor{red}{4}, 9_1] \rightarrow A = [\textcolor{blue}{1}, \textcolor{blue}{4}, 9_2, 9_1]$



$i=3$   $A = [\textcolor{blue}{1}, \textcolor{blue}{4}, \textcolor{red}{9_2}, 9_1] \rightarrow A = [\textcolor{blue}{1}, \textcolor{blue}{4}, \textcolor{blue}{9_2}, 9_1]$

Resultado:  $A = [\textcolor{blue}{1}, \textcolor{blue}{4}, \textcolor{blue}{9_2}, \textcolor{blue}{9_1}]$

No  
estable



# Método de INSERCIÓN

Es un algoritmo casi tan sencillo como el de selección pero mas flexible.

Es el método que se utiliza a menudo en un juego de cartas para ordenar las cartas que se tienen en la mano a medida que las recibe las va intercalando entre las ya ordenadas.

El método consiste en el análisis sucesivo del arreglo de elementos y de la inserción de cada uno de ellos en el lugar que corresponda en los anteriores ordenados.

# Método de INSERCIÓN

La iteración se construye de modo que:

**Antes del paso k:**

$\overbrace{a_1 \leq a_2 \leq a_3 \leq \dots \leq a_{k-1}}^{\text{ordenados}}, \overbrace{a_k, a_{k+1}, \dots, a_n}^{\text{desordenados}}$

**Paso k:** *insertar*  $a_k$  en el lugar apropiado en  $(a_1, \dots, a_{k-1})$  desplazando los que corresponda a la derecha.

**Después del paso k:**

$\overbrace{a_1 \leq a_2 \leq a_3 \leq \dots \leq a_{k-1} \leq a_k}^{\text{ordenados}}, \overbrace{a_{k+1}, \dots, a_n}^{\text{desordenados}}$

# Algoritmo de INSERCIÓN

Algoritmo **Inserción**(A,n)

**Entrada:** A: arreglo de (1.. MAX) elementos de tipoitem  
n: nro. entero de datos a ordenar

**Salida:** A arreglo ordenado.

Auxiliar: i,j:entero;      proximo: tipoitem

```
PARA i = 2 ,n HACER
    proximo ← A(i)
    j ← i
    MIENTRAS ( j > 1  AND  proximo < A(j-1) ) HACER
        A(j) ← A(j-1)
        j ← j-1
    A(j) ← proximo
FIN
```



# Método de INSERCIÓN

Ej. Ordenar el arreglo de 15 letras: ASORTINGEXAMPLE

A	S	O	R	T	I	N	G	E	X	A	M	P	L	E									
A	S	O	R	T	I	N	G	E	X	A	M	P	L	E									
A	O	S	R	T	I	N	G	E	X	A	M	P	L	E									
A	O	R	S	T	I	N	G	E	X	A	M	P	L	E									
A	O	R	S	T	I	N	G	E	X	A	M	P	L	E									
A	I	O	R	S	T	I	N	G	E	X	A	M	P	L	E								
A	I	N	O	R	S	T	I	N	G	E	X	A	M	P	L	E							
A	G	I	N	O	R	S	T	I	N	G	E	X	A	M	P	L	E						
A	E	G	I	N	O	R	S	T	I	N	G	E	X	A	M	P	L	E					
A	E	G	I	N	O	R	S	T	I	N	G	E	X	A	M	P	L	E					
A	A	E	G	I	N	O	R	S	T	I	N	G	E	X	A	M	P	L	E				
A	A	E	G	I	M	N	O	R	S	T	I	N	G	E	X	A	M	P	L	E			
A	A	E	G	I	M	N	O	P	R	S	T	I	N	G	E	X	A	M	P	L	E		
A	A	E	G	I	L	M	N	O	P	R	S	T	I	N	G	E	X	A	M	P	L	E	
A	A	E	E	G	I	L	M	N	O	P	R	S	T	I	N	G	E	X	A	M	P	L	E
A	A	E	E	G	I	L	M	N	O	P	R	S	T	I	N	G	E	X	A	M	P	L	E

Arreglo original

i=2	j=2
i=3	j=3,2
i=4	j=4,3
i=5	j=5
i=6	j=6,5,4,3,2
i=7	j=7,6,5,4,3
i=8	j=8,7,6,5,4,3,2
i=9	j=9, 8,7,6,5,4,3,2
i=10	j=10
i=11	j=11,10,9, 8,7,6,5,4,3,2
i=12	j=12,11,10,9, 8,7,6
i=13	j=13, 12,11,10,9
i=14	j=14,13,12,11,10,9,8,7,6
i=15	j=15,14,13, 12,11,10,9,8,7,6,5,4

Arreglo ordenado



# Propiedades

## Algoritmo de INSERCIÓN

**Tiempo de Ejecución:**  $T(n) \in O(n^2)$

**Memoria:** almacenamiento  $\in O(1)$  de espacio extra.

**Sensibilidad:** muy sensible, disminuye a  $O(n)$  cuando el arreglo está ordenado.

**Estabilidad:** SI es estable

**Otras:** por su condición de estable y de muy sensible se usa para finalizar el ordenamiento cuando se aplica algoritmos recursivos.

# Método de INTERCAMBIO

Este método se basa en la comparación sucesiva e *intercambio de elementos adyacentes* del arreglo hasta que quede ordenado.

El algoritmo es una iteración, en cada paso se va comparando, empezando por el final, cada elemento con el anterior y en caso de estar desordenados se intercambian.

De esa forma en el primer paso quedará el elemento menor en la primera posición del arreglo, en la segunda iteración quedara el segundo menor en la segunda posición y así continúa ubicando los mas pequeños.

Por ello también se lo conoce con el nombre de *método de la burbuja* (bubble sort) ya que los elementos mas pequeños van avanzando rápidamente a las primeras posiciones del arreglo.

# Algoritmo de INTERCAMBIO

Algoritmo **Intercambio**(A,n)

**Entrada:** A: arreglo de (1.. MAX) elementos de tipoitem  
n: nro. entero de datos a ordenar

**Salida:** A arreglo ordenado.

Auxiliar: i,j:entero; aux: tipoitem

```
PARA i = 2 , n HACER
    PARA j = n, i paso -1 HACER
        SI A(j-1) > A(j) ENTONCES
            aux ← A(j-1)
            A(j-1) ← A(j)
            A(j) ← aux
```

FIN

# Método de INTERCAMBIO

Ej. Ordenar el arreglo de 15 letras: ASORTINGEXAMPLE

A	S	O	R	T	I	N	G	E	X	A	M	P	L	E
A	A	S	O	R	T	I	N	G	E	X	E	M	P	L
A	A	E	S	O	R	T	I	N	G	E	X	L	M	P
A	A	E	E	S	O	R	T	I	N	G	L	X	M	P
A	A	E	E	G	S	O	R	T	I	N	L	M	X	P
A	A	E	E	G	I	S	O	R	T	L	N	M	P	X
A	A	E	E	G	I	L	S	O	R	T	M	N	P	X
A	A	E	E	G	I	L	M	S	O	R	T	N	P	X
A	A	E	E	G	I	L	M	N	S	O	R	T	P	X
A	A	E	E	G	I	L	M	N	O	S	P	R	T	X
A	A	E	E	G	I	L	M	N	O	P	S	R	T	X
A	A	E	E	G	I	L	M	N	O	P	R	S	T	X
A	A	E	E	G	I	L	M	N	O	P	R	S	T	X
A	A	E	E	G	I	L	M	N	O	P	R	S	T	X
A	A	E	E	G	I	L	M	N	O	P	R	S	T	X

Arreglo original

i=2	j=15,14,...2
i=3	j=15,14,...3
i=4	j=15,14,...4
i=5	j=15,14,...5
i=6	j=15,14,...6
i=7	j=15,14,...7
i=8	j=15,14,...8
i=9	j=15,14,...9
i=10	j=15,14,...10
i=11	j=15,14,...11
i=12	j=15,14,...12
i=13	j=15,14,13
i=14	j=15,14
i=15	j=15

Arreglo ordenado

# Propiedades

## Algoritmo de INTERCAMBIO

**Tiempo de Ejecución:**  $T(n) \in O(n^2)$

**Memoria:** almacenamiento  $\in O(1)$  de espacio extra.

**Sensibilidad:** muy sensible

**Estabilidad:** SI es estable

**Otras:** es el método simple que mas movimientos hace en el peor caso.

# Comparación de los métodos simples de ordenación

METODO	mínimo	medio	máximo
SELECCION	$C=(n^2-n)/2$ $M=3(n-1)$	$C=(n^2-n)/2$ $M=n(\ln n+0.57)$	$C=(n^2-n)/2$ $M=n^2/4+3(n-1)$
INSERCIÓN	$C=n-1$ $M=2(n-1)$	$C=(n^2+n-2)/4$ $M=(n^2+9n-10)/4$	$C=(n^2-n)/2-1$ $M=(n^2+3n-4)/2$
INTERCAMBIO	$C=(n^2-n)/2$ $M=0$	$C=(n^2-n)/2$ $M=3(n^2-n)/4$	$C=(n^2-n)/2$ $M=3(n^2-n)/2$

# Limitaciones de los métodos simples de Ordenación Interna

Un análisis más profundo, junto con los resultados experimentales muestran que selección e intercambio son muy ineficientes para muchos datos y no recomendables más que por su simplicidad.

El método de inserción es también ineficiente para ordenar con propósito general, pero su costo general baja para arreglos pequeños y parcialmente ordenados.

El tamaño de la entrada será un factor clave cuando  $n$  sea grande en la elección del método de ordenación.



# Métodos mejorados de Ordenación Interna

Existen algunos métodos mas eficientes para la ordenación de vectores, basados en los mecanismos de los métodos simples.

Estos **métodos mejorados** se pueden utilizar cuando:

- Cuando  $n$  sea grande.
- Cuando el método tiene que ejecutarse muy a menudo.
- Cuando se necesite una rápida ejecución de la tarea.