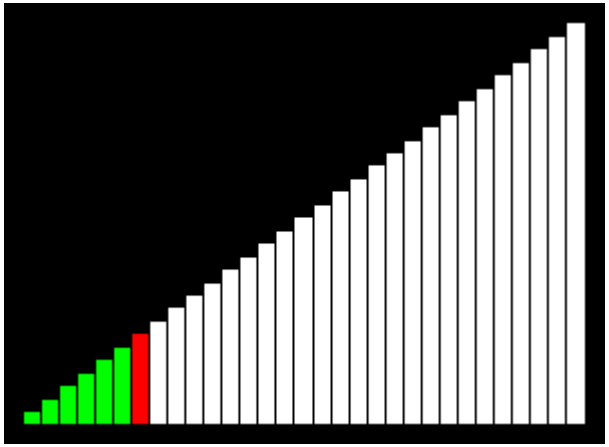
Algoritmos y Estructuras de Datos I

Facultad de Ciencias Exactas y Tecnología
Universidad Nacional de Tucumán

2023

UNIVERSIDAD NACIONAL DE TUCUMÁN
facet
FACULTAD DE CIENCIAS EXACTAS Y TECNOLOGÍA

Ordenación(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.

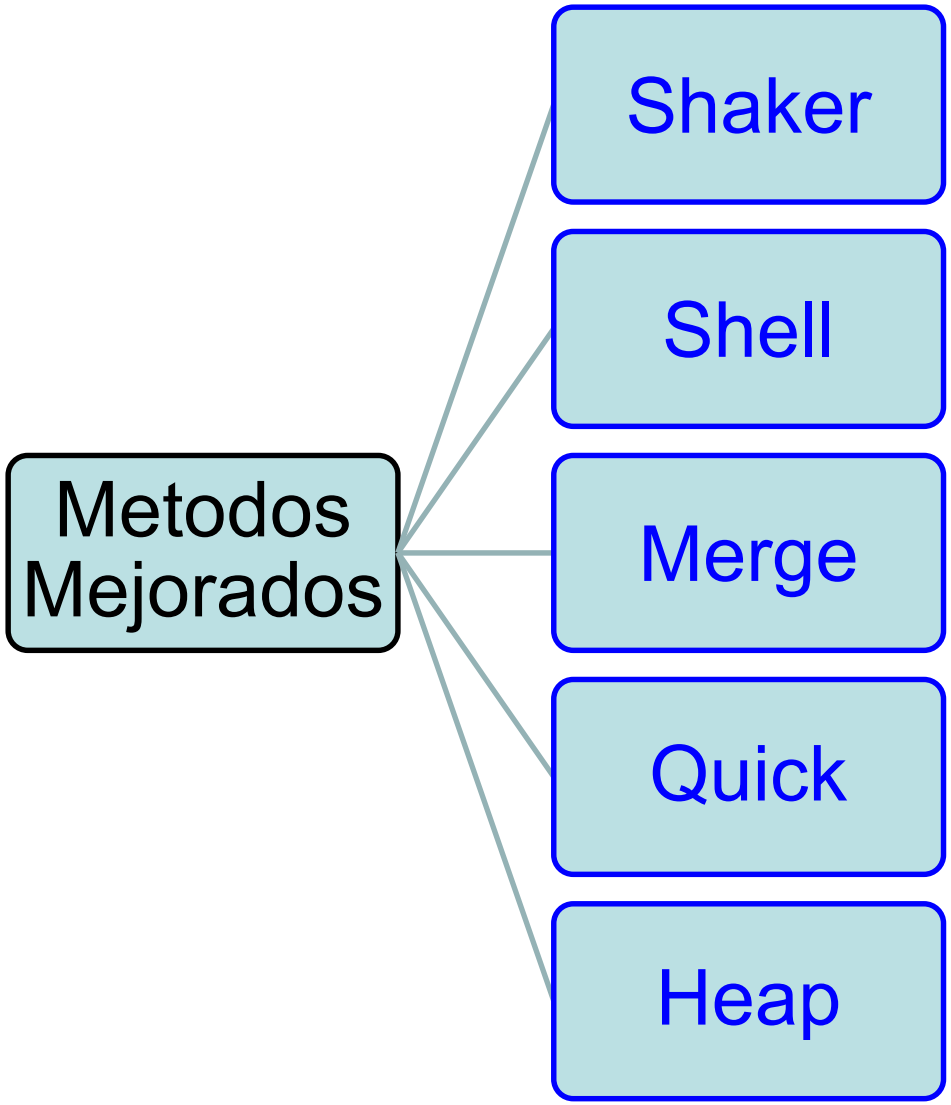
Métodos mejorados de Ordenación Interna

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.

Métodos de Ordenación Interna



Método SHAKER

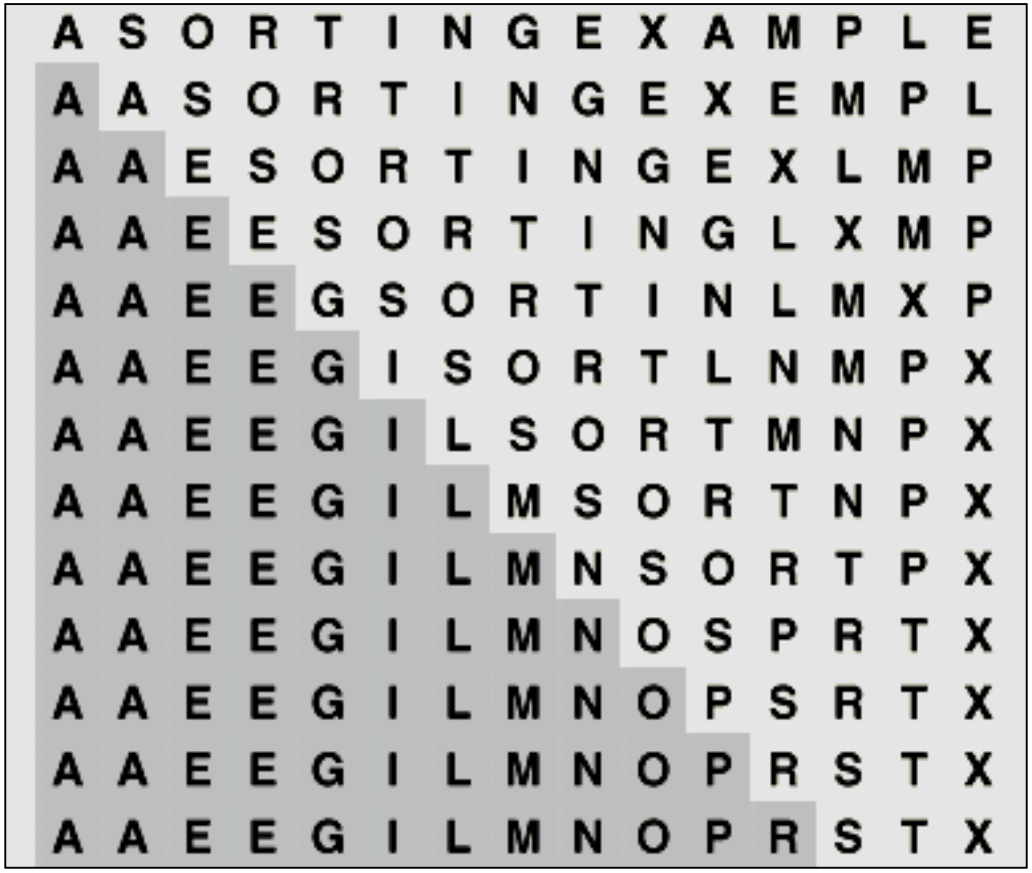
El metodo **shaker sort** tambien se conoce con el nombre en inglés de **cocktail sort** o de la **burbuja bidireccional** y es una mejora del método de la burbuja.

El algoritmo de la burbuja admite fácilmente algunas mejoras:

- A veces las ultimas pasadas no tienen efecto, porque el arreglo está ya ordenado. Una técnica obvia de mejorar es ***controlar si se ha producido un intercambio en una pasada***. Si no se hizo intercambio el arreglo ya está ordenado.
- Se puede controlar no solo si se ha producido un intercambio sino ***almacenar la posición (índice) del último intercambio***. Es claro que todos los ítems por debajo de este índice ya están ordenados.

Método Shaker

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

Arreglo original

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

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

Arreglo ordenado

Método SHAKER

Ejemplo: n=8

A = [12 18 42 55 66 94 44 06] Arreglo original

A = [**06 12** 18 42 55 66 94 44] i=2, j=8,7,…2 ind=2

A = [**06 12 18 42 44 55** 66 94] i=3, j=8,7,…3 ind=6

A = [**06 12 18 42 44 55 66 94**] i=4, j=8,…,6 ind=8

A = [**06 12 18 42 44 55 66 94**] Arreglo ordenado

Método SHAKER

Hay una asimetría particular en la burbuja. Un ítem liviano a la derecha burbujea en una sola pasada. Un ítem pesado necesita n-1 pasadas para llegar a su posición definitiva.

Así por ejemplo el arreglo:

A = [12 18 42 44 55 67 94 **06**]

se ordena en 1 sola pasada.

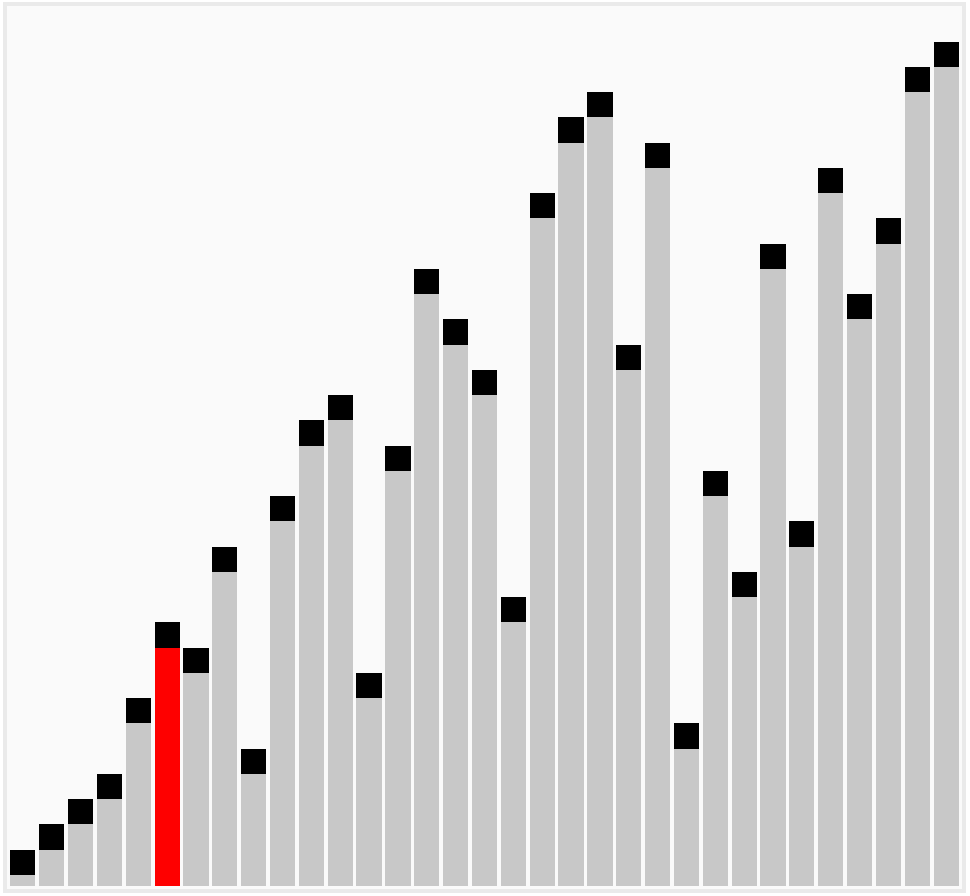
En cambio el arreglo:

B = [**96** 06 12 18 42 44 55 67]

necesita 7 pasadas.

- Esto sugiere otra mejora, ***alternar las direcciones de las pasadas consecutivas***.

Método SHAKER



Propiedades
Algoritmo SHAKER

Tiempo de Ejecución:

- Mejor Caso : $T(n) \in O(n)$
- Caso medio: $T(n) \in O(n^2)$
- Peor Caso: $T(n) \in O(n^2)$

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

Sensibilidad: muy muy sensible

Estabilidad: SI es estable

Otras:Todas estas mejoras no reducen el número de intercambios (que es lo más costoso) sino de comparaciones redundantes, por lo tanto las mejoras tendrán un efecto mucho menos profundo del esperado.

Método SHELL

Método de Intercambios Decrecientes

Propuesto por Donald Shell en 1959 fue uno de los primeros algoritmos de ordenación que rompió la barrera cuadrática del tiempo.

Es una versión mas eficiente del método de Inserción y del de Intercambio.

El método trabaja comparando elementos que están distantes; la distancia entre comparaciones decrece con las iteraciones, hasta que en la ultima pasada, se comparan elementos adyacentes.

Shell sugiere una secuencia de incrementos decrecientes: $h_N,\ldots,h_2,h_1,h_0$

Algunas secuencias son mejores que otras pero en todas $h_0=1$.

Método SHELL

Secuencia h ordenada u ordenada de h en h

Definición: una secuencia se dice que está ***h-ordenada***, si al considerar cualquier subsecuencia de elementos separados h posiciones, estos se encuentran ordenados.

Una secuencia h ordenada se puede pensar como h subsecuencias ordenadas que se encuentran encajadas.

Definición: se llama ***h-ordenamiento*** al proceso de tomar una secuencia y dejarla h-ordenada.

Teorema: si una secuencia está **k-ordenada** y se la somete a un **h-ordenamiento**, entonces seguirá k-ordenada.

Método SHELL

Secuencia *h ordenada*

Ejemplo:

La secuencia:

S= (27, 15, 30, 01, 35, 20, 31, 05, 36, 28, 32, 41, 40, 39, 33, 42)

No está ordenada. Pero si está 4 ordenada:

S= (27, 15, 30, 01, 35, 20, 31, 05, 36, 28, 32, 41, 40, 39, 33, 42)

S=	27	15	30	01	35	20	31	05	36	28	32	41	40	39	33	42
S1=	27				35				36				40			
S2=		15				20				28				39		
S3=			30				31				32				33	
S4=				01				05				41				42

Método SHELL

D. Shell propuso el siguiente método:

- Se deja la secuencia h_N-ordenada.
- Se hace otra pasada tomando $h_{N-1}<h_N$ y se la deja h_{N-1} ordenada.
- Se sigue así hasta $h_0=1$ que es la ultima pasada.

- En 1971, James Peterson y David Russel en la Universidad de Stanford, con gran numero de experimentos determinaron que las mejores secuencias son de la forma:

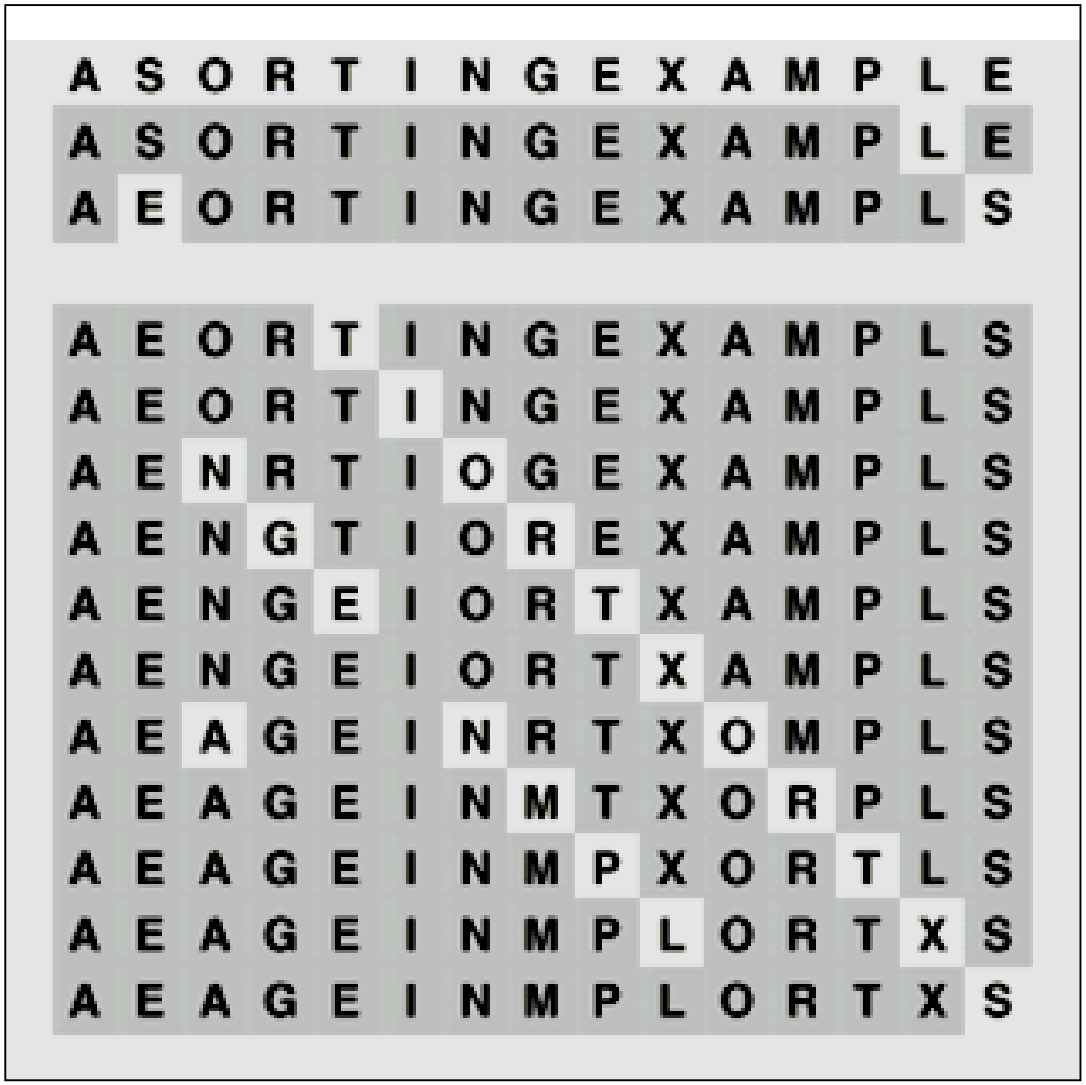
secuencia	
$1,3,5,9,\ldots,2^k+1$	
$1,3,7,15,\ldots,2^k-1$	$h_i=2h_{i-1}+1$, $h_0=1$, Wirth
$1,3,5,11,\ldots,(2^k+1)/3$ ó $(2^k-1)/3$	
$1,4,13,40,\ldots, (3^k-1)/2$	$h_i=3h_{i-1}+1$, $h_0=1$, Knuth

Método SHELL

```
Algoritmo ShellSort(A,n)
 Entrada: A: arreglo de (1.. MAX) elementos de tipoitem
          n: nro. entero de datos a ordenar
Salida: A arreglo ordenado.
Auxiliar:  h,i,j:entero; proximo: tipoitem
h ←1
REPETIR    h ← h*3+1    //genera secuencia de Knuth
      HASTA QUE  h > n
REPETIR
          h ← h / 3
          PARA i= h+1 ,n HACER
                    proximo ← A(i) ;  j ← i
                    MIENTRAS (j>h) AND (proximo < A(j-h)) HACER
                              A(j) ← A(j-h) ;     j ← j-h
                    A(j) ← proximo
HASTA QUE h =1
FIN
```

Método SHELL

Ej. Ordenar el arreglo de 15 letras: ASORTINGEXAMPLE



h=13 (primera pasada)

i=14 j=14

i=15 j=15,2

h=4 (segunda pasada)

i=5 j=5

i=6 j=6

i=7 j=7,3

i=8 j=8,4

i=9 j=9,5

i=10 j=10

i=11 j=11,7,3

i=12 j=12,8

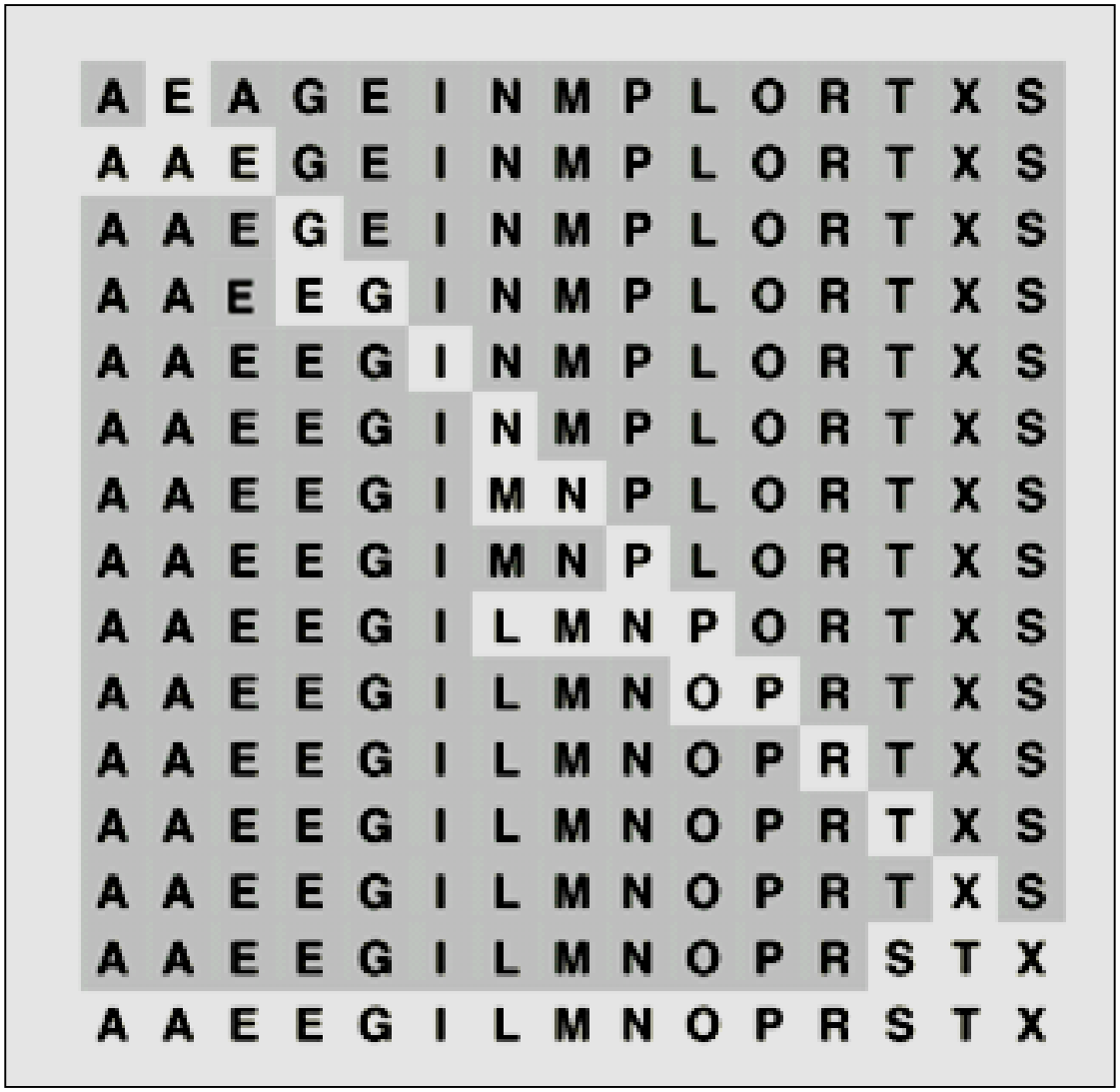
i=13 j=13,9

i=14 j=14,10

i=15 j=15

Método SHELL

Ej. Ordenar el arreglo de 15 letras: ASORTINGEXAMPLE



h=1 (ultima pasada)

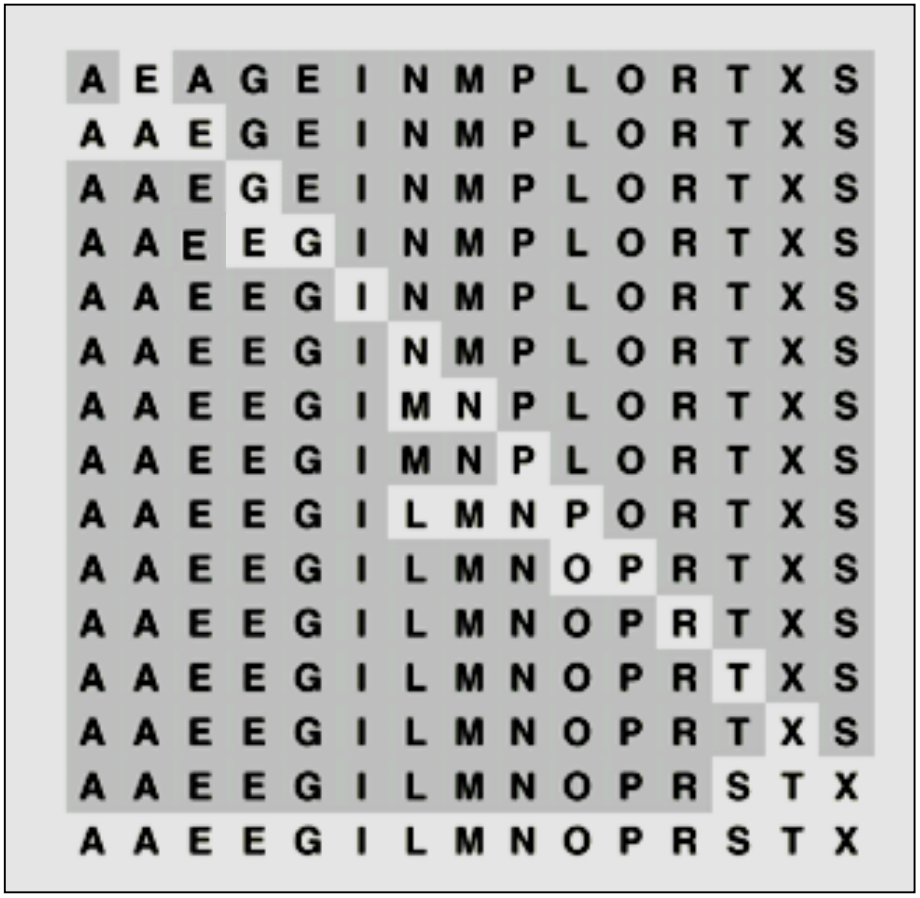
i=2	j=2
i=3	j=3,2,1
i=4	j=4
i=5	j=5,4
i=6	j=6
i=7	j=7
i=8	j=8,7
i=9	j=9
i=10	j=10,9,8,7
i=11	j=11,10
i=12	j=12
i=13	j=13
i=14	j=14
i=15	j=15,14,13

ordenado

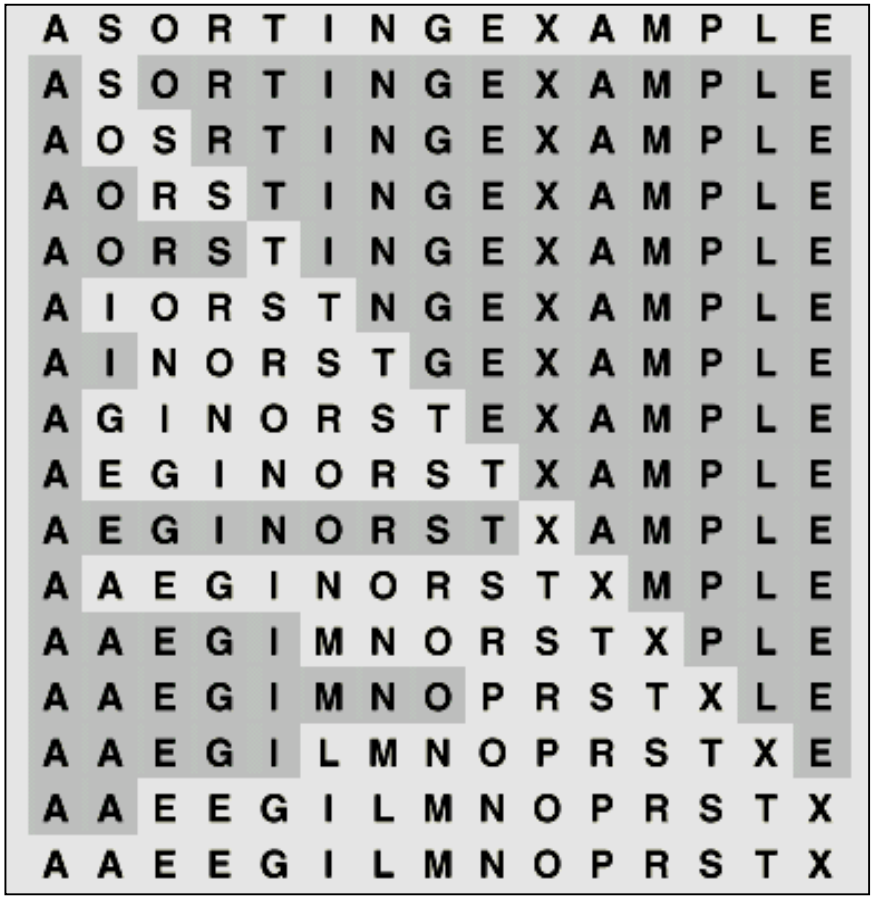
Método SHELL vs INSERCION

Ej. Ordenar el arreglo de 15 letras: ASORTINGEXAMPLE

Shell h=1 (ultima pasada)



Insercion



Propiedades
Algoritmo de SHELL

Tiempo de Ejecución: todos los tiempos de ejecución son empíricos:

Con secuencia de Knuth $T(n) \in O(n^{3/2})$.

Para otras secuencias $T(n) \in O(n^{4/3})$ y hasta $T(n) \in O(n \cdot \log^2(n))$.

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

Sensibilidad: es sensible, la complejidad disminuye a $O(n \cdot \log(n))$ en el caso de arreglos ordenados.

Estabilidad: NO es estable

Método MERGE o de MEZCLA

La ordenación por mezcla es un prototipo de la estrategia Divide & Conquer y se puede implementar en forma recursiva y en forma iterativa.

La idea principal de este método de ordenación es:

- Si el vector tiene tamaño n=0 o n=1 está ordenado (caso base), sino dividir el vector de tamaño n en dos partes iguales de n/2 elementos cada una.
- Ordenar cada subvector
- Hacer una mezcla de los dos subvectores ordenados para obtener el resultado final.

Método de MEZCLA

El proceso de ***mezcla*** o ***fusión*** permite **combinar** dos arreglos ordenados en otro más grande, también ordenado.

Ejemplo:

A1=[1 , 2 , 3 , 5 , 7 , 8 , 9]

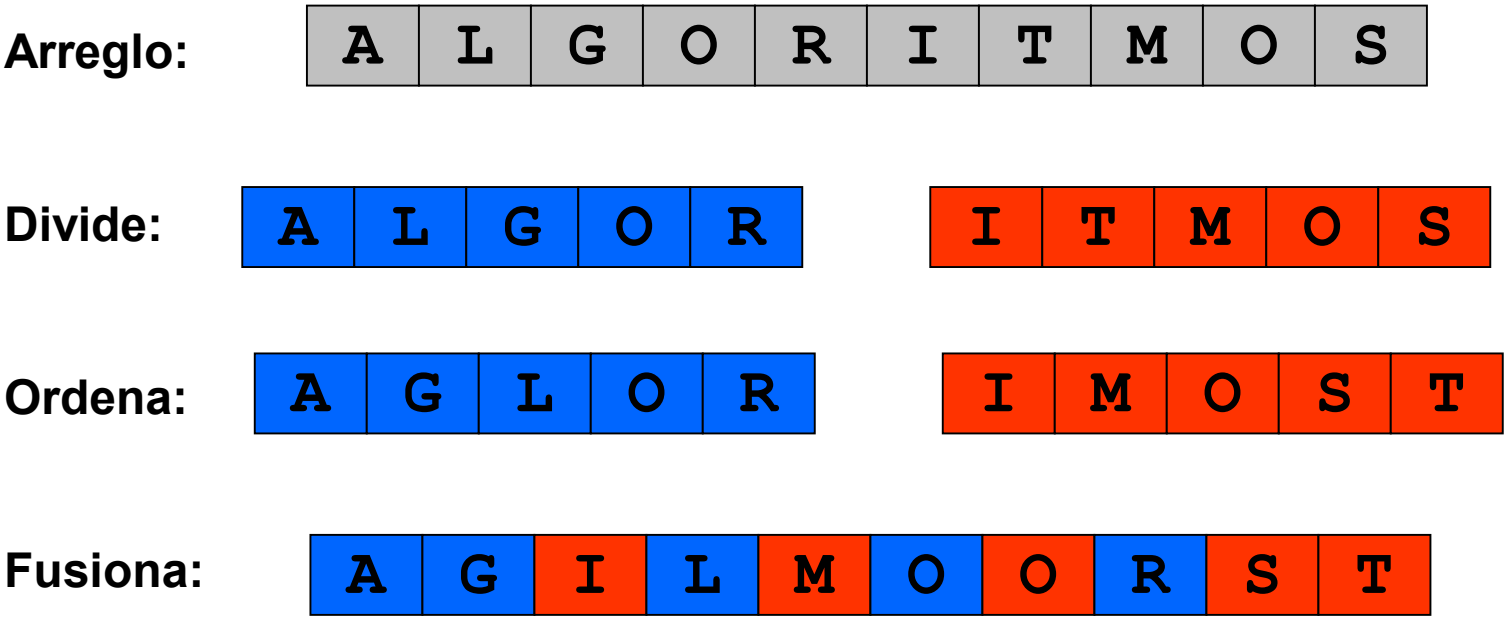
A2=[2 , 4 , 6 , 8]

Combinar A1 y A2:

A=[1 , 2 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 8 , 9]

Método de MEZCLA

Ejemplo:



Método de MEZCLA

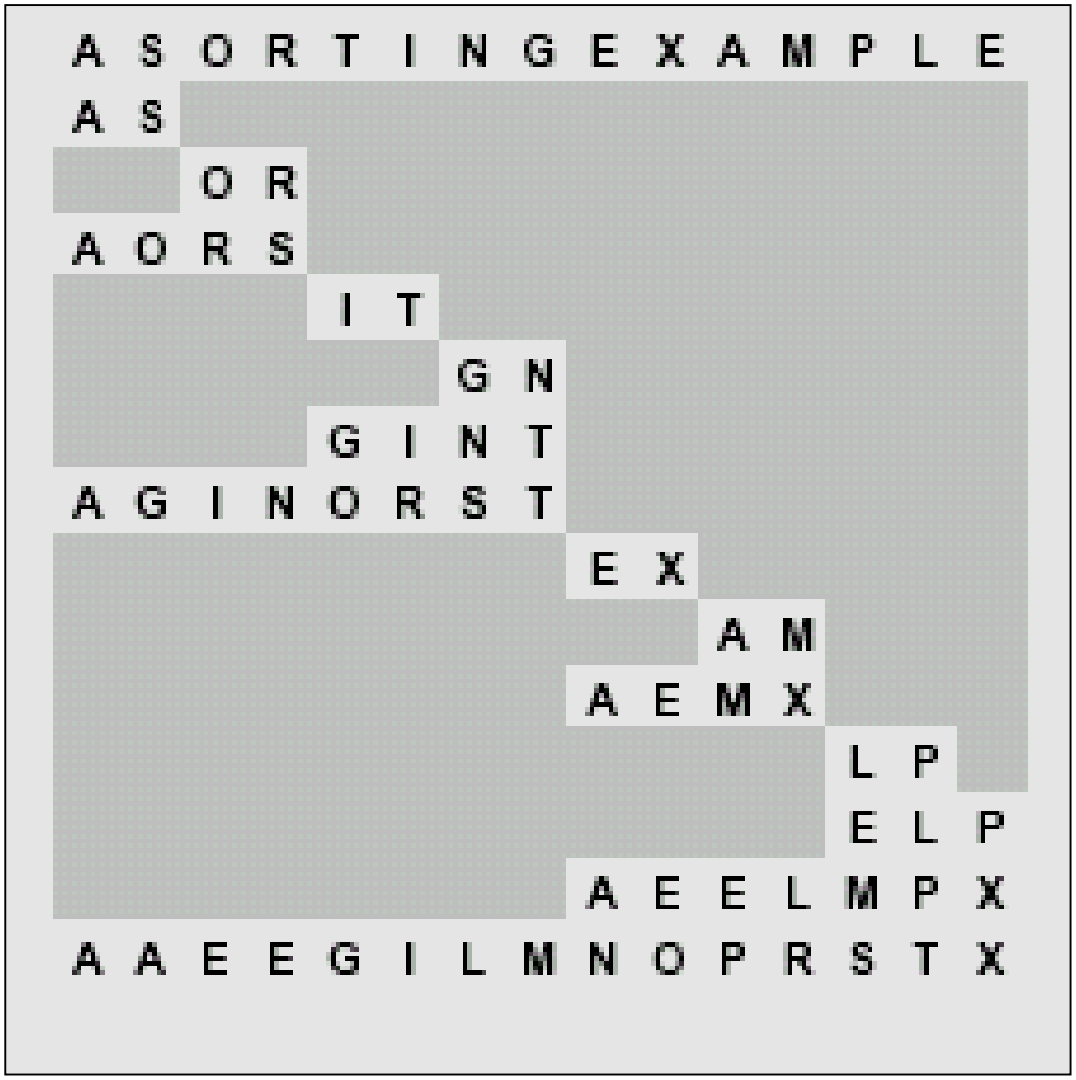
Algoritmo Mezcla(A,izq,der)
Entrada: A: arreglo de (1.. MAX) elementos de tipoitem
izq,der: indices entre los que se aplicará la clasificacion.
Salida: A arreglo ordenado entre los indices izq y der.
auxiliar: medio $\in$ entero

```
SI der > izq ENTONCES
        medio = (der+izq) / 2
        Mezcla (A , izq , medio)
        Mezcla (A , medio+1 , der)
        Combinar(A [izq,medio] , A [medio+1,der] )
 FIN
```

Se invoca con Mezcla (A,1,n)

Método de MEZCLA

Ejemplo



Propiedades
Algoritmo de MEZCLA

Tiempo de Ejecución: $T(n) \epsilon O(n \cdot \log_2 n)$.

Memoria: almacenamiento ϵ $O(n)$ de espacio extra para combinar y espacio extra para las pilas de las llamadas recursivas.

Sensibilidad: nada sensible.

Estabilidad: SI es estable

Método de MEZCLA

VENTAJAS:

- Ordena un arreglo de n elementos en un tiempo proporcional a O(n.logn) aun en el peor caso.
- Es el único método de costo ϵ O(n·log(n)) que es estable.
- Se puede implementar de forma que se pueda acceder secuencialmente a los datos, lo que a veces resulta una ventaja.
- Es el método ideal para ordenar una lista enlazada, en la que el acceso secuencial es la única forma posible de acceso.
- Es el método adecuado para combinar dos archivos ordenados y formar un tercer archivo ordenado.

DESVENTAJA:

Su principal inconveniente es que cuando se ordena un arreglo no se puede evitar la utilización de un espacio extra proporcional a n.

Método QUICKSORT o RAPIDO

Fue presentado en 1962 en los Computer Journal por C.A.R. Hoare y es un método muy difundido en la actualidad. Hoare llamó “Quicksort” a su método, un nombre muy apropiado, ya que en la práctica supera en eficiencia a los otros métodos.

Es un método de ordenamiento por intercambio de una partición, basado en el método de Intercambio Directo.

Quicksort usa la estrategia Divide & Conquer, ya que se ordena por partición, es decir dividiendo el arreglo en dos partes más pequeñas, cada una de las cuales se ordena en forma independiente de la misma forma.

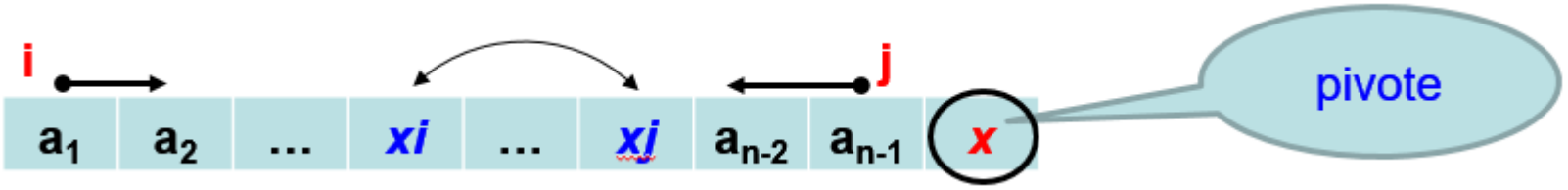
Método QUICKSORT
Partición

La idea de la partición es la siguiente:

Tomar arbitrariamente un elemento ***x*** del arreglo A llamado **pivote**.

Repetir los siguientes pasos hasta que los índices se encuentren:

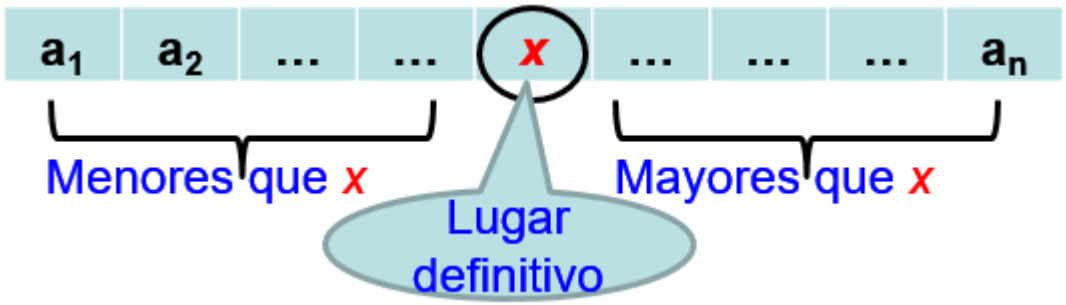
- Inspeccionar el arreglo de izquierda a derecha hasta encontrar un ***elemento mayor que x***, sea ***xi***.
- Inspeccionar el arreglo de derecha a izquierda hasta encontrar un ***elemento menor que x***, sea ***xj***.
- ***Intercambiar xi*** con **xj**.



Método QUICKSORT
Partición

La iteración termina cuando los índices se encuentren en el medio del arreglo.

Llevar el pivote a esa posición mediante un ultimo intercambio.

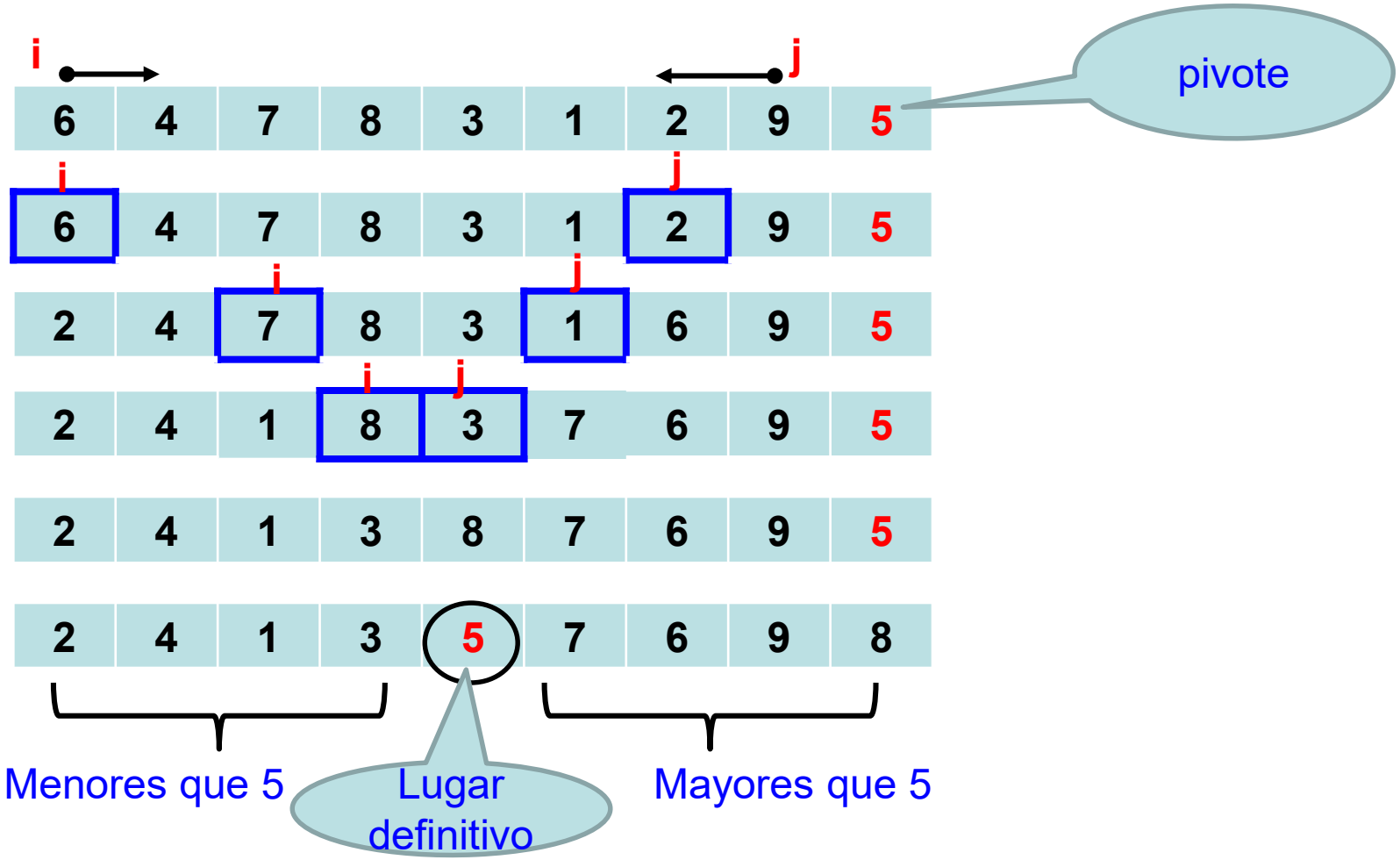


Después de la partición:

- El elemento ***x*** está en su posición definitiva en el arreglo ordenado.
- Todos los elementos ubicados a su izquierda son menores que él
- Todos los elementos a su derecha son mayores que él.

Método QUICKSORT
Partición

Ejemplo: Partición con el último elemento del arreglo.



Método QUICKSORT
Partición

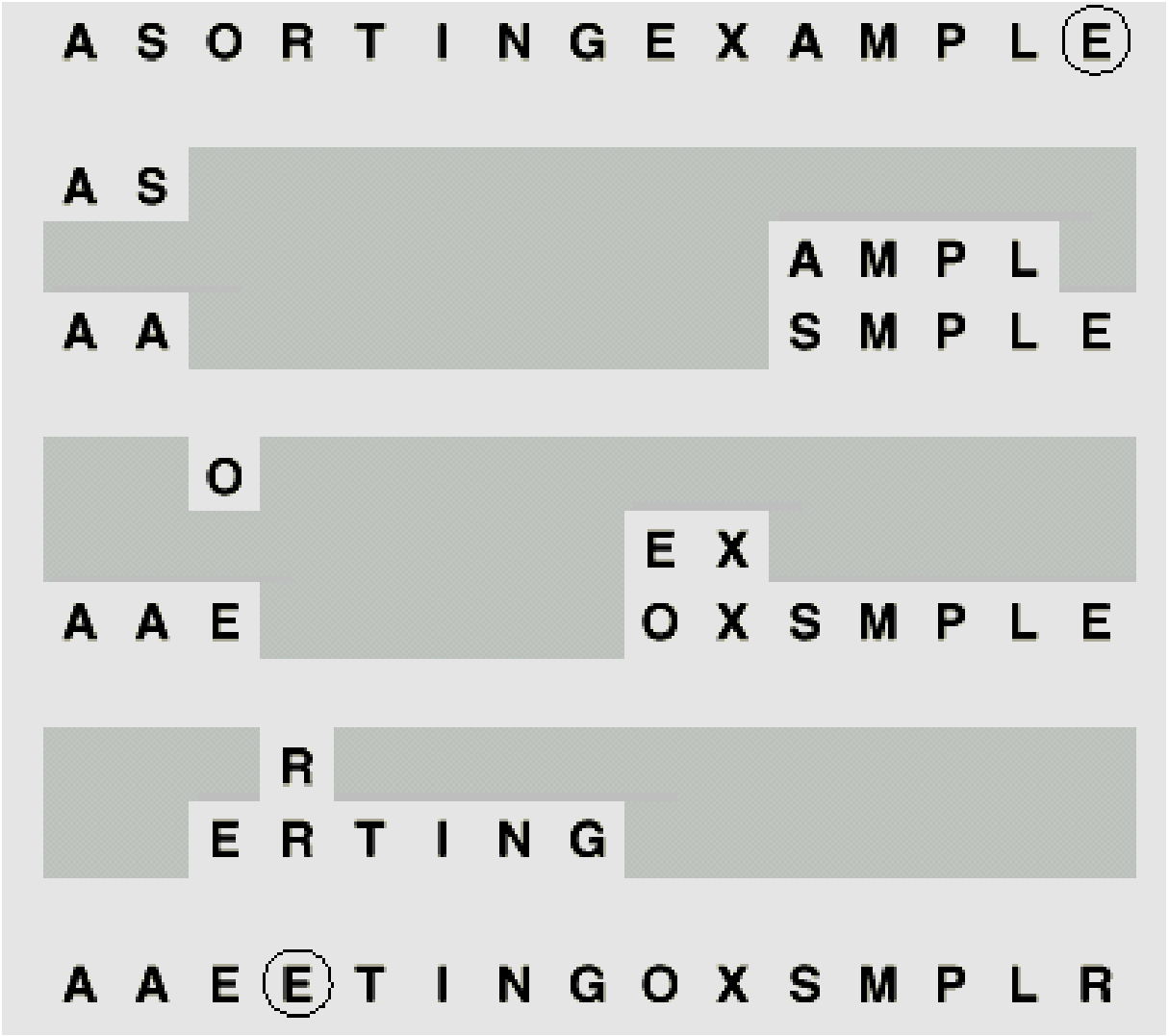
Función UnaPartición (A,n)
Entrada: A: arreglo de (1.. MAX) elementos de tipoitem
n: nro. entero de elementos de A
Salida: indice entero donde se ubica el pivote de la partición.
A arreglo con la partición elemento indice
Auxiliares: x, aux ∈ tipoitem; i,j ∈ entero

```
x ←A(n)      // se usa el ultimo elemento como pivote de la particion
i ← 0 ; j ← n
REPETIR
        REPETIR  i ← i+1   MIENTRAS  A(i) < x
        REPETIR  j ← j-1   MIENTRAS  A(j) > x
        SI i < j ENTONCES
                aux ← A(i) ; A(i) ← A(j) ; A(j) ← aux
MIENTRAS  i < j
aux ← A(i) ; A(i) ← A(n) ; A(n) ← aux ;
Retorna i
FIN
```

T(n) ϵ O(n)

Método QUICKSORT

Ej. de partición respecto de la letra E marcada



E pivote

i=2

j=11

Intercambio S con A

i=3

j=9

Intercambio O con E

i=4

j=3 → se cruzan

Intercambio R con E

Arreglo particionado

Por el pivote E

Método QUICKSORT

Algoritmo QuickSort (A,izq,der)

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

izq,der: indices enteros entre los que se aplicará la clasificacion.

Salida: A arreglo ordenado entre los indices izq y der.

auxiliar: indice $\in$ entero

```
SI der > izq ENTONCES
        indice ← Partición(A , izq , der)
        QuickSort (A , izq , indice-1)
        QuickSort  (A , indice+1 , der)
FIN
```

Se invoca con QuickSort (A,1,n)

Método QUICKSORT
Partición

Función Partición (A,izq,der) : arreglo x entero → entero

Entrada: A: arreglo de (1.. MAX) elementos de tipoitem
izq, der : índices enteros entre los cuales se hace la partición

Salida: índice entero donde se ubica el pivote de la partición.
A arreglo con la partición entre índices izq y der

Retorna: índice entero donde se ubica el pivote de la partición.

Auxiliares: x, aux $\in$ tipoitem; i,j $\in$ entero

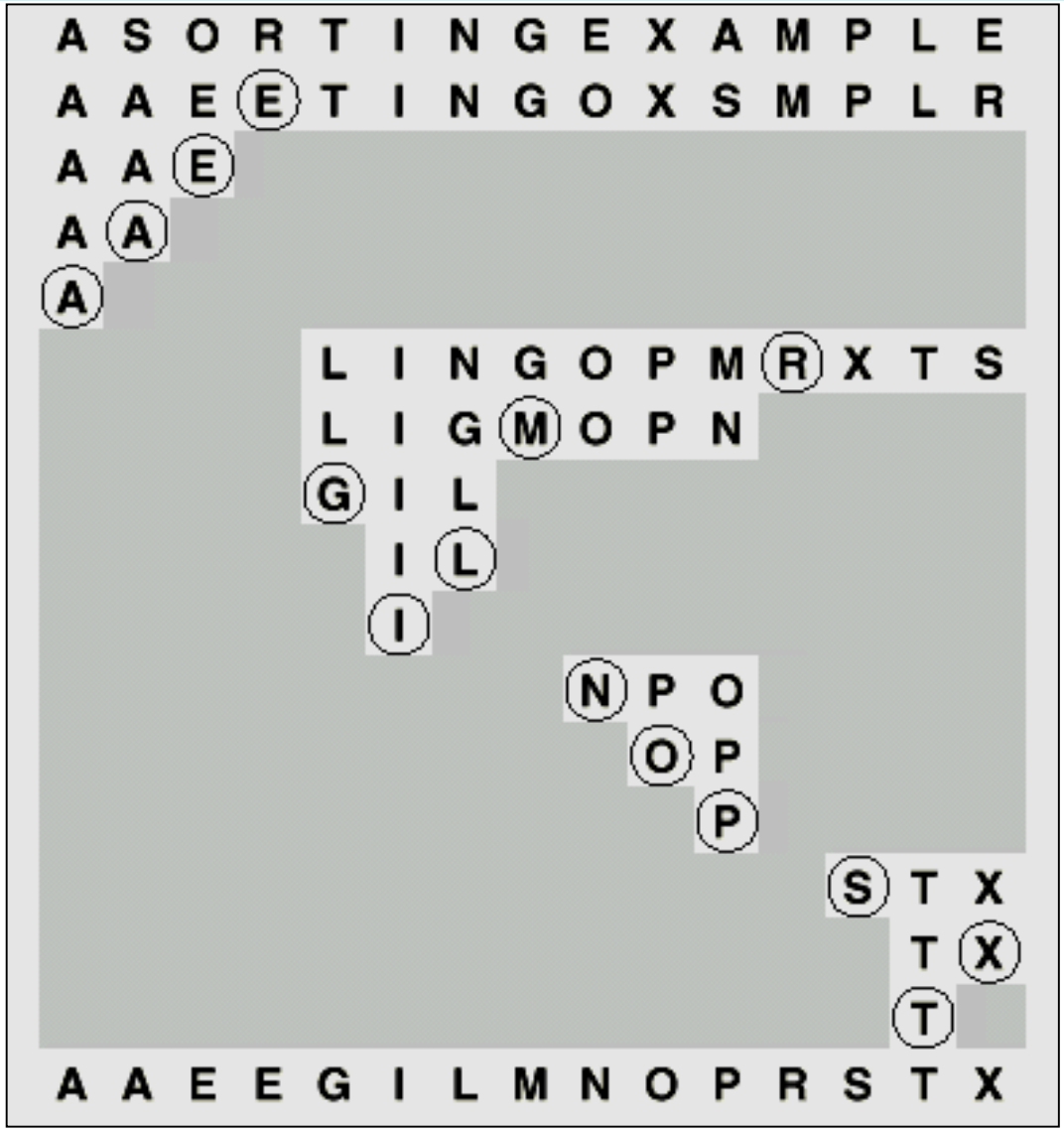
Método QUICKSORT
Partición

Función Partición (A,izq,der) : arreglo x entero → entero

```
x ← A(der);      i ← izq-1 ;      j ← der
REPETIR
        REPETIR  i ←i+1   MIENTRAS  A(i) < x
        REPETIR  j ←j-1   MIENTRAS  A(j) > x
        SI i < j ENTONCES
                aux ← A(i) ;      A(i) ←A(j);      A(j) ← aux
MIENTRAS  i < j
aux ← A(i)
A(i) ← A(der)
A(der) ← aux
Retorna i
FIN
```

Método QUICKSORT

Ej. tomando como pivote el ultimo elemento del arreglo



Arreglo original

QuickSort (A,1,15) , índice=4

QuickSort (A,1,3) , índice=3

QuickSort (A,1,2) , índice=2

QuickSort (A,1,1)

QuickSort (A,5,15) , índice=12

QuickSort (A,5,11) , índice=8

QuickSort (A,5,7) , índice=5

QuickSort (A,6,7) , índice=7

QuickSort (A,6,6)

QuickSort (A,9,11), índice=9

QuickSort (A,10,11), índice=10

QuickSort (A,11,11)

QuickSort (A,13,15), índice=13

QuickSort (A,14,15), índice=15

QuickSort (A,14,14)

Arreglo Ordenado

Método QUICKSORT

Análisis del costo de tiempo

Caso mas favorable: en cada etapa de partición se divide el arreglo exactamente por la mitad (descomposición con balance). En este caso se puede analizar el tiempo con la siguiente forma recurrente:

$T(n)=c1$ si $n=1$

$T(n)=2\,T(n/2) + c2\,n$ si $n>1$, donde c1 y c2 son constantes.

Desarrollando esta forma se puede concluir que en este caso :

$$T(n) \text{ es } O(n \log_2 n)$$

Caso medio:

Se puede probar que en el caso medio $T(n)=1.386\ n \log_2 n$, que es el mismo crecimiento que tiene el mejor caso.

Método QUICKSORT

Análisis del costo de tiempo

Peor Caso:

El caso mas desafortunado es que se tome un pivote correspondiente a un extremo del arreglo, por ejemplo el mayor. En ese caso un segmento de n elementos de divide en n-1 ítems a la izquierda y 1 solo item a la derecha. Si esto ocurre en cada paso se precisan n procesos de subdivisión en lugar de $\log_2 n$ y produce el rendimiento mas desfavorable del método.

$T(n)=c1$ si $n=1$

$T(n)=T(n-1) + c2\ n$ si $n>1$ donde c1 y c2 son constantes.

Desarrollando esta forma se puede concluir que en este caso :

$T(n)$ es $O(n^2)$

Método QUICKSORT

Evidentemente el paso crucial es la ***selección del elemento pivote*** en cada paso de partición

- **Mejor Caso :** $T(n) \in O(n \log_2 n)$
- **Caso medio:** $T(n) \in O(n \log_2 n)$
- **Peor Caso:** $T(n) \in O(n^2)$

Como alejarse del peor caso?

Con una buena elección del pivote:

- Mediana del arreglo
- Mediana de 3 del arreglo

Otra mejora: usar otro algoritmo de ordenación para partes pequeñas. No seguir aplicando la recursión para tramos del arreglo de pocos elementos, inserción directa es un método mas rápido de ordenación.

Propiedades
Algoritmo de QUICKSORT

Tiempo de Ejecución:

Mejor caso: $T(n) \in O(n \cdot \log_2 n))$.
Peor Caso: $T(n) \in O(n^2)$

Memoria: almacenamiento $\in O(n)$ de espacio extra para las pilas de las llamadas recursivas.

Sensibilidad: algo sensible.

Estabilidad: NO es estable

Otras: su buen rendimiento disminuye mucho cuando se encuentra con grupos de claves iguales. Para estos casos se plantea otro algoritmo que hace la particion en 3 secciones del arreglo.