

INF 263 – Algoritmia

Fuerza Bruta

CAPITULO II - ESTRATEGIAS

FUERZA BRUTA

A solid orange horizontal bar spanning the width of the slide, located at the bottom.

Fuerza Bruta



Fuerza Bruta

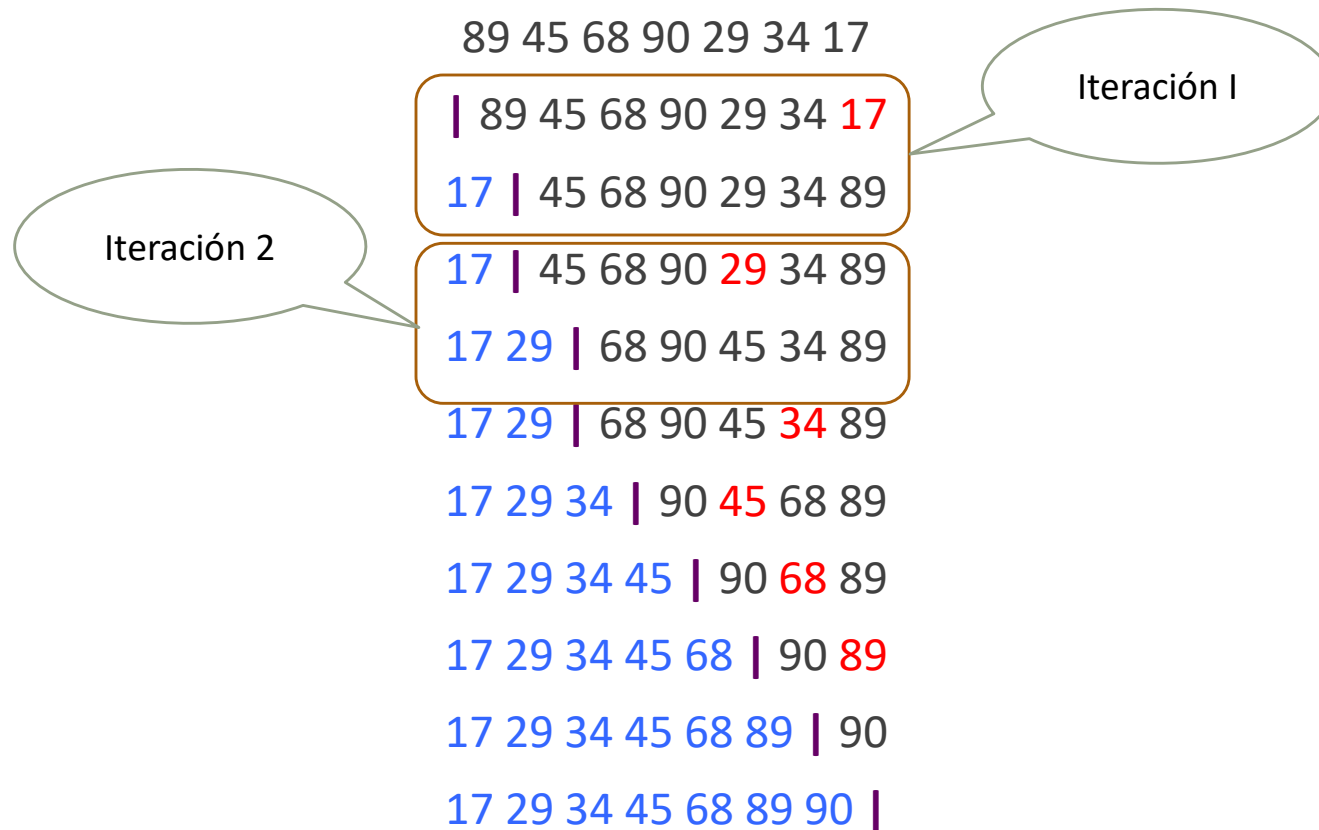
- Método directo para resolver un problema.
- Método fácil de aplicar, que no implica mucho razonamiento por parte de diseñador.
- Aplicar pasos básicos basándose en la definición del problema.
- Es la estrategia por defecto que se nos ocurriría a la mayoría para resolver un problema.

Aplicación de la Estrategia

Ordenación

Ordenación - Selección

- Definir una secuencia de pasos para ordenar la secuencia



Ordenación por Selección

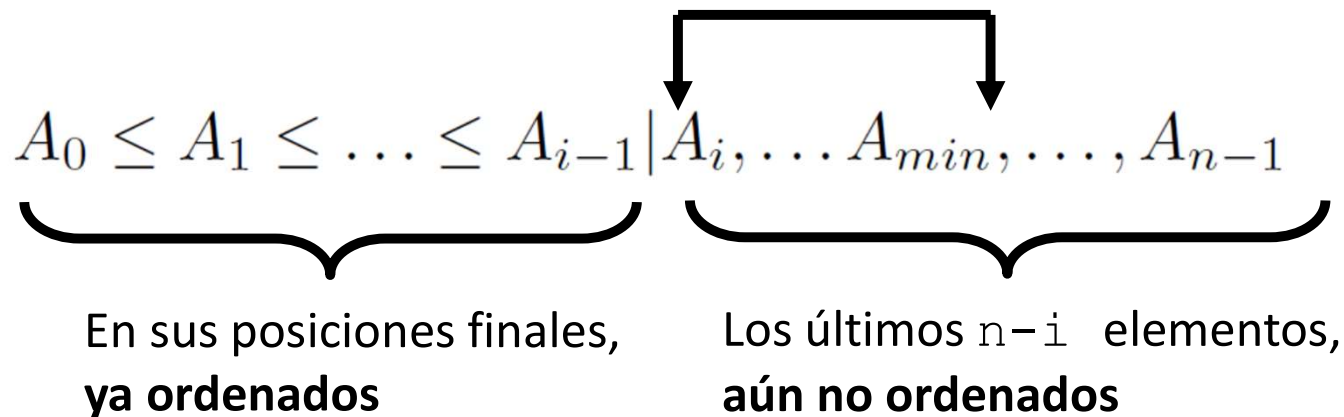
Estrategia

- Encontrar el menor elemento e intercambiarlo por el primer elemento
 - Primer elemento ya está ordenado
- Encontrar el menor elemento de los $n - 1$ elementos restantes e intercambiarlo por el segundo elemento
 - Segundo elemento ya está ordenado
- Repetir proceso con todos los elementos que todavía no están en posición final (ordenados).

Ordenación por Selección

Estrategia

- En el i -ésimo paso ($0 \leq i \leq n-2$), se busca el elemento más pequeño de la lista $[i..n-1]$ y se intercambia con el elemento de la posición i



Ordenación por Selección

Pseudocódigo

Precondición: Arreglo $A[0..n-1]$ desordenado

Postcondición: Arreglo $A[0..n-1]$ ordenado

```
Para  $i \leftarrow 0$  hasta  $n-2$  hacer
     $\text{min} \leftarrow i$ 
    Para  $j \leftarrow i+1$  hasta  $n-1$  hacer
        Si  $A[j] < A[\text{min}]$  entonces
             $\text{min} \leftarrow j$ 
        Fin Si
    Fin Para
    intercambiar  $A[i]$  y  $A[\text{min}]$ 
Fin Para
```

Ordenación por Selección

Pseudocódigo

Precondición: Arreglo $A[0..n-1]$ desordenado

Postcondición: Arreglo $A[0..n-1]$ ordenado

Para $i \leftarrow 0$ **hasta** $n-2$ **hacer**

$\text{min} \leftarrow i$

Para $j \leftarrow i+1$ **hasta** $n-1$ **hacer**

Si $A[j] < A[\text{min}]$ **entonces**

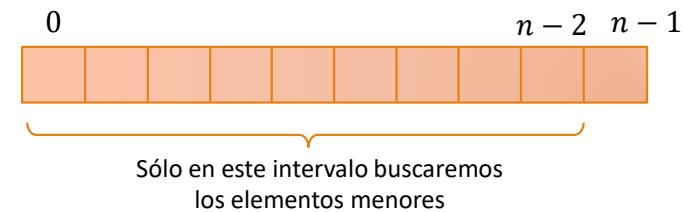
$\text{min} \leftarrow j$

Fin Si

Fin Para

intercambiar $A[i]$ y $A[\text{min}]$

Fin Para



Ordenación por Selección

Pseudocódigo

Precondición: Arreglo $A[0..n-1]$ desordenado

Postcondición: Arreglo $A[0..n-1]$ ordenado

Para $i \leftarrow 0$ **hasta** $n-2$ **hacer**

$\text{min} \leftarrow i$

Para $j \leftarrow i+1$ **hasta** $n-1$ **hacer**

Si $A[j] < A[\text{min}]$ **entonces**

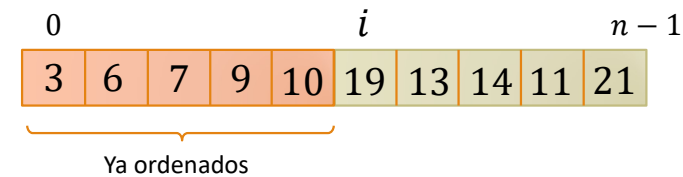
$\text{min} \leftarrow j$

Fin Si

Fin Para

intercambiar $A[i]$ y $A[\text{min}]$

Fin Para



Buscar el menor elemento de la secuencia entre los índices i y $n-1$

Ordenación por Selección

Pseudocódigo

Precondición: Arreglo $A[0..n-1]$ desordenado

Postcondición: Arreglo $A[0..n-1]$ ordenado

Para $i \leftarrow 0$ hasta $n-2$ hacer

```
min ← i
```

Para $j \leftarrow i+1$ hasta $n-1$ hacer

Si $A[j] < A[\text{min}]$ entonces

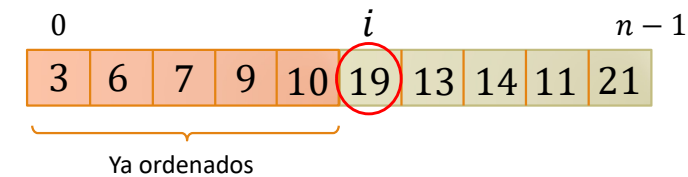
$$\text{min} \leftarrow j$$

Fin Si

Fin Para

intercambiar $A[i]$ y $A[\text{min}]$

Fin Para



Asumimos que el i -ésimo elemento es el mínimo

Ordenación por Selección

Pseudocódigo

Precondición: Arreglo $A[0..n-1]$ desordenado

Postcondición: Arreglo $A[0..n-1]$ ordenado

Para $i \leftarrow 0$ hasta $n-2$ hacer

```
min ← i
```

Para $j \leftarrow i+1$ hasta $n-1$ hacer 

Si $A[j] < A[\text{min}]$ entonces

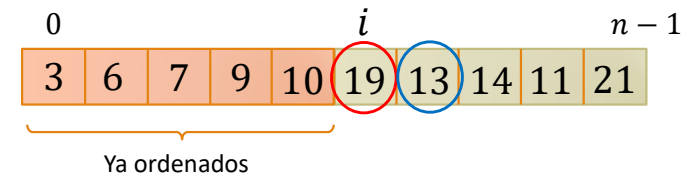
$$\text{min} \leftarrow j$$

Fin Si

Fin Para

intercambiar $A[i]$ y $A[\text{min}]$

Fin Para



Iteramos desde el $i + 1$ -ésimo elemento
haciendo la comparación por el mínimo

Ordenación por Selección

Pseudocódigo

Precondición: Arreglo $A[0..n-1]$ desordenado

Postcondición: Arreglo $A[0..n-1]$ ordenado

Para $i \leftarrow 0$ hasta $n-2$ hacer

$$\text{min} \leftarrow i$$

Para $j \leftarrow i+1$ hasta $n-1$ hacer

Si $A[j] < A[\text{min}]$ entonces

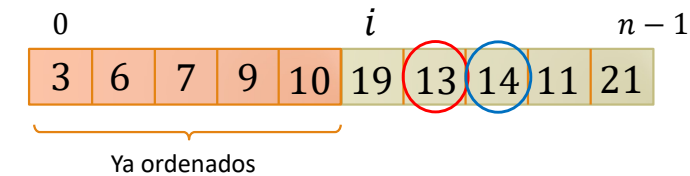
$$\text{min} \leftarrow j$$

Fin Si

Fin Para

intercambiar $A[i]$ y $A[\text{min}]$

Fin Para



Seguimos buscando un mínimo

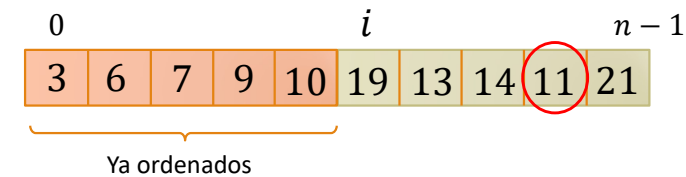
Ordenación por Selección

Pseudocódigo

Precondición: Arreglo $A[0..n-1]$ desordenado

Postcondición: Arreglo $A[0..n-1]$ ordenado

```
Para  $i \leftarrow 0$  hasta  $n-2$  hacer  
     $\text{min} \leftarrow i$   
    Para  $j \leftarrow i+1$  hasta  $n-1$  hacer  
        Si  $A[j] < A[\text{min}]$  entonces  
             $\text{min} \leftarrow j$   
        Fin Si  
    Fin Para  
    intercambiar  $A[i]$  y  $A[\text{min}]$   
Fin Para
```



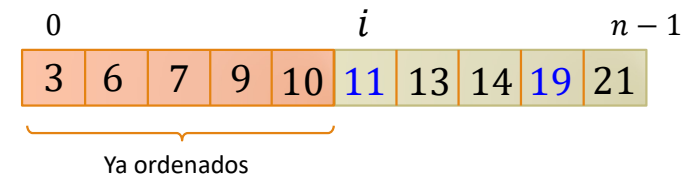
Ordenación por Selección

Pseudocódigo

Precondición: Arreglo $A[0..n-1]$ desordenado

Postcondición: Arreglo $A[0..n-1]$ ordenado

```
Para  $i \leftarrow 0$  hasta  $n-2$  hacer  
     $\text{min} \leftarrow i$   
    Para  $j \leftarrow i+1$  hasta  $n-1$  hacer  
        Si  $A[j] < A[\text{min}]$  entonces  
             $\text{min} \leftarrow j$   
        Fin Si  
    Fin Para  
    intercambiar  $A[i]$  y  $A[\text{min}]$   
Fin Para
```



Intercambiamos los valores

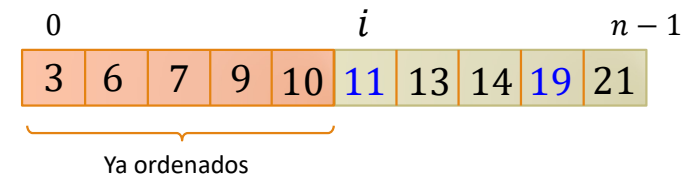
Ordenación por Selección

Pseudocódigo

Precondición: Arreglo $A[0..n-1]$ desordenado

Postcondición: Arreglo $A[0..n-1]$ ordenado

```
Para  $i \leftarrow 0$  hasta  $n-2$  hacer  
     $\text{min} \leftarrow i$   
    Para  $j \leftarrow i+1$  hasta  $n-1$  hacer  
        Si  $A[j] < A[\text{min}]$  entonces  
             $\text{min} \leftarrow j$   
        Fin Si  
    Fin Para  
    intercambiar  $A[i]$  y  $A[\text{min}]$   
Fin Para
```



Al término de la i -ésima iteración, se intercambian el mínimo elemento con el elemento en la posición i .

Ordenación - Burbuja

Iteración 1:

89 45 68 90 29 34 17

“Se van comparando progresivamente los elementos en pares, haciendo que cumplan con el orden requerido”

Ordenación - Burbuja

Iteración 1:

89 45 68 90 29 34 17

89 \leftrightarrow ? 45 68 90 29 34 17

Se comparan los dos primeros elementos

Si no cumplen con el orden, se intercambian

Ordenación - Burbuja

Iteración 1:

89 45 68 90 29 34 17

89 \leftrightarrow ? 45 68 90 29 34 17

45 89 \leftrightarrow ? 68 90 29 34 17

Se vuelven a comparar los elementos de la misma forma

Ordenación - Burbuja

Iteración 1:

Al finalizar la i -ésima iteración
El último elemento está ordenado

89 45 68 90 29 34 17
89 $\leftrightarrow?$ 45 68 90 29 34 17
45 89 $\leftrightarrow?$ 68 90 29 34 17
45 68 89 $\leftrightarrow?$ 90 29 34 17
45 68 89 29 90 $\leftrightarrow?$ 34 17
45 68 89 29 34 90 $\leftrightarrow?$ 17
45 68 89 29 34 17 | 90

Ordenación - Burbuja

Iteración 2:


45 68 89 29 34 17 | 90
45 $\leftrightarrow?$ 68 89 29 34 17 | 90
45 68 $\leftrightarrow?$ 89 29 34 17 | 90
45 68 89 $\leftrightarrow?$ 29 34 17 | 90
45 68 29 89 $\leftrightarrow?$ 34 17 | 90
45 68 29 34 89 $\leftrightarrow?$ 17 | 90
45 68 29 34 17 | 89 90

Ordenación - Burbuja

- Comparar elementos adyacentes e intercambiarlos si están fuera de orden.
- El mayor elemento se va como una “burbuja” hasta su posición final.
- Si se repite este procedimiento para la secuencia $[0..n-2]$, al final el segundo mayor elemento se irá como una “burbuja” hasta su posición final.
- Después de $n - 1$ iteraciones, la secuencia está ordenada.

Ordenación - Burbuja

- El i -ésimo paso ($0 \leq i \leq n-2$) se puede representar de la siguiente manera

$$A_0, \dots, A_j \Leftrightarrow^? A_{j+1}, \dots, A_{n-i-1} \mid A_{n-i} \leq \dots \leq A_{n-1}$$


Los primeros $n-i$ elementos,
aún no ordenados


En sus posiciones
finales, **ya ordenados**

Ordenación Burbuja

Pseudocódigo

Precondición: Arreglo $A[0..n-1]$ desordenado

Postcondición: Arreglo $A[0..n-1]$ ordenado

Para $i \leftarrow 0$ **hasta** $n-2$ **hacer**  **Intuición:** Es necesario hacer $n - 1$ iteraciones para resolver el problema

Para $j \leftarrow 0$ **hasta** $n-2-i$ **hacer**

Si $A[j+1] < A[j]$ **entonces**

 intercambiar $A[j]$ y $A[j+1]$

Fin Si

Fin Para

Fin Para

Ordenación Burbuja

Pseudocódigo

Precondición: Arreglo $A[0..n-1]$ desordenado

Postcondición: Arreglo $A[0..n-1]$ ordenado

Para $i \leftarrow 0$ **hasta** $n-2$ **hacer**

Para $j \leftarrow 0$ **hasta** $n-2-i$ **hacer**

Si $A[j+1] < A[j]$ **entonces**

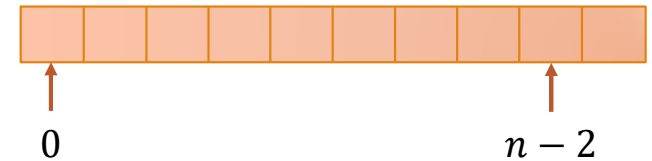
intercambiar $A[j]$ y $A[j+1]$

Fin Si

Fin Para

Fin Para

Intuición: Es necesario hacer $n - 1$ iteraciones para resolver el problema



Cuando $i = 0$, intervalo de comparación $[0, n - 2]$

Ordenación Burbuja

Pseudocódigo

Precondición: Arreglo $A[0..n-1]$ desordenado

Postcondición: Arreglo $A[0..n-1]$ ordenado

Para $i \leftarrow 0$ **hasta** $n-2$ **hacer**

Para $j \leftarrow 0$ **hasta** $n-2-i$ **hacer**

Si $A[j+1] < A[j]$ **entonces**

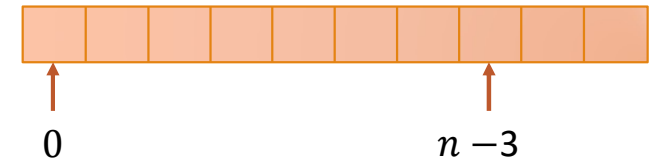
intercambiar $A[j]$ y $A[j+1]$

Fin Si

Fin Para

Fin Para

Intuición: Es necesario hacer $n - 1$ iteraciones para resolver el problema



Cuando $i = 0$, intervalo de comparación $[0, n - 2]$

Cuando $i = 1$, intervalo de comparación $[0, n - 3]$

Ordenación Burbuja

Pseudocódigo

Precondición: Arreglo $A[0..n-1]$ desordenado

Postcondición: Arreglo $A[0..n-1]$ ordenado

Para $i \leftarrow 0$ **hasta** $n-2$ **hacer**

Para $j \leftarrow 0$ **hasta** $n-2-i$ **hacer**

Si $A[j+1] < A[j]$ **entonces**

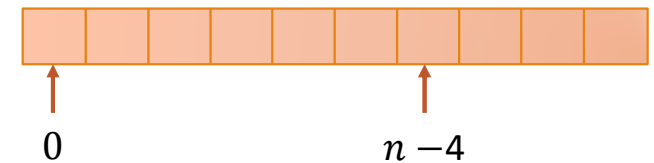
intercambiar $A[j]$ y $A[j+1]$

Fin Si

Fin Para

Fin Para

Intuición: Es necesario hacer $n - 1$ iteraciones para resolver el problema



Cuando $i = 0$, intervalo de comparación $[0, n - 2]$

Cuando $i = 1$, intervalo de comparación $[0, n - 3]$

Cuando $i = 2$, intervalo de comparación $[0, n - 4]$

Ordenación Burbuja

Pseudocódigo

Precondición: Arreglo $A[0..n-1]$ desordenado

Postcondición: Arreglo $A[0..n-1]$ ordenado

Para $i \leftarrow 0$ **hasta** $n-2$ **hacer**

Para $j \leftarrow 0$ **hasta** $n-2-i$ **hacer**

Si $A[j+1] < A[j]$ **entonces**

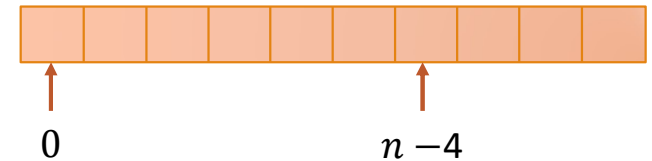
intercambiar $A[j]$ y $A[j+1]$

Fin Si

Fin Para

Fin Para

Intuición: Es necesario hacer $n - 1$ iteraciones para resolver el problema



Cuando $i = 0$, intervalo de comparación $[0, n - 2]$

Cuando $i = 1$, intervalo de comparación $[0, n - 3]$

Cuando $i = 2$, intervalo de comparación $[0, n - 4]$

Para cualquier i , intervalo de comparación $[0, n - 2 - i]$

Ordenación Burbuja

Pseudocódigo

Precondición: Arreglo $A[0..n-1]$ desordenado

Postcondición: Arreglo $A[0..n-1]$ ordenado

Para $i \leftarrow 0$ **hasta** $n-2$ **hacer**

Para $j \leftarrow 0$ **hasta** $n-2-i$ **hacer**

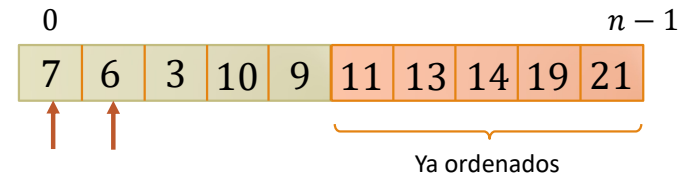
Si $A[j+1] < A[j]$ **entonces** ←
 intercambiar $A[j]$ y $A[j+1]$

Fin Si

Fin Para

Fin Para

Ejemplo en iteración 5



Comparar elementos, como no guarda orden, se intercambian

Ordenación Burbuja

Pseudocódigo

Precondición: Arreglo $A[0..n-1]$ desordenado

Postcondición: Arreglo $A[0..n-1]$ ordenado

Para $i \leftarrow 0$ **hasta** $n-2$ **hacer**

Para $j \leftarrow 0$ **hasta** $n-2-i$ **hacer**

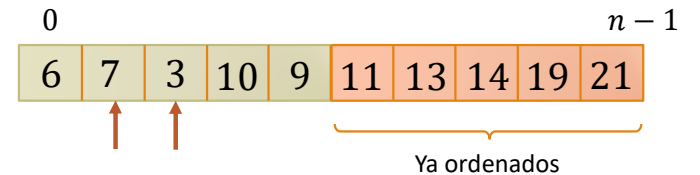
Si $A[j+1] < A[j]$ **entonces**
 intercambiar $A[j]$ y $A[j+1]$

Fin Si

Fin Para

Fin Para

Ejemplo en iteración 5



Comparar elementos, como no guarda orden, se intercambian

Ordenación Burbuja

Pseudocódigo

Precondición: Arreglo $A[0..n-1]$ desordenado

Postcondición: Arreglo $A[0..n-1]$ ordenado

Para $i \leftarrow 0$ **hasta** $n-2$ **hacer**

Para $j \leftarrow 0$ **hasta** $n-2-i$ **hacer**

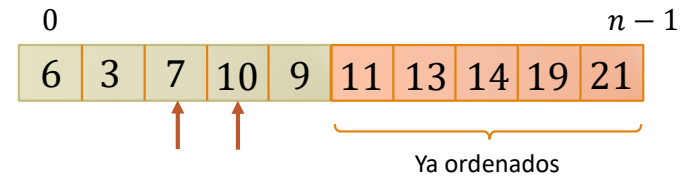
Si $A[j+1] < A[j]$ **entonces**
 intercambiar $A[j]$ y $A[j+1]$

Fin Si

Fin Para

Fin Para

Ejemplo en iteración 5



Comparar elementos, como guardan orden, no pasa nada

Ordenación Burbuja

Pseudocódigo

Precondición: Arreglo $A[0..n-1]$ desordenado

Postcondición: Arreglo $A[0..n-1]$ ordenado

Para $i \leftarrow 0$ **hasta** $n-2$ **hacer**

Para $j \leftarrow 0$ **hasta** $n-2-i$ **hacer**

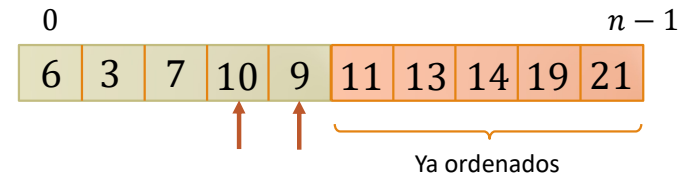
Si $A[j+1] < A[j]$ **entonces** ←
 intercambiar $A[j]$ y $A[j+1]$

Fin Si

Fin Para

Fin Para

Ejemplo en iteración 5



Comparar elementos, como no guardan orden, se intercambian

Ordenación Burbuja

Pseudocódigo

Precondición: Arreglo $A[0..n-1]$ desordenado

Postcondición: Arreglo $A[0..n-1]$ ordenado

Para $i \leftarrow 0$ **hasta** $n-2$ **hacer**

Para $j \leftarrow 0$ **hasta** $n-2-i$ **hacer**

Si $A[j+1] < A[j]$ **entonces**

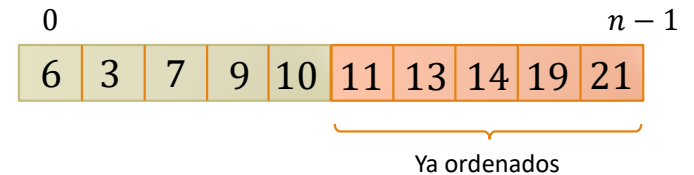
intercambiar $A[j]$ y $A[j+1]$

Fin Si

Fin Para

Fin Para

Ejemplo en iteración 5



Termina la iteración y ahora tenemos un elemento más ordenado

Aplicación de la Estrategia

Más ejemplos



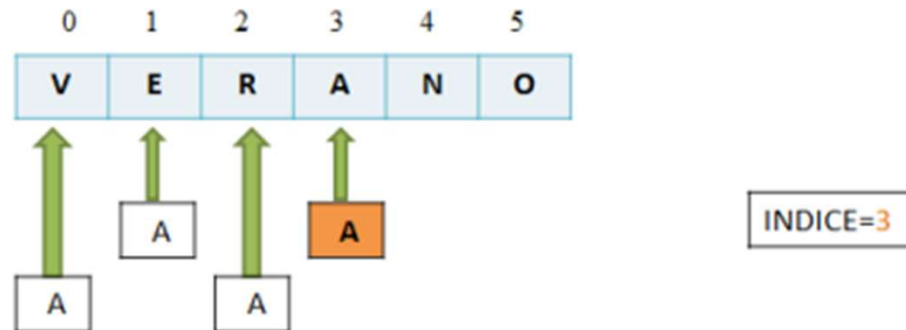
Búsqueda - Secuencial

- Dado un conjunto de valores, encontrar un elemento dado.
- Existen varios algoritmos
 - Algunos más rápidos requieren memoria adicional
 - Algunos muy rápidos requieren que los elementos estén ordenados
- Con el uso de estructuras de datos (última parte de nuestro curso), podemos tener algoritmos rápidos para hacer búsquedas.

Búsqueda Secuencial

Estrategia

- Comparar el elemento buscado con cada elemento de la colección.
- Posibles resultados:
 - Se encuentra la clave (búsqueda con éxito)
 - No existen más elementos para comparar (búsqueda con fracaso)



Búsqueda Secuencial

Pseudocódigo

Precondición: Arreglo $A[0..n-1]$ y una llave de búsqueda K

Postcondición: El índice del primer elemento de A que coincide con la llave K o -1 si la llave no se encuentra

$i \leftarrow 0$

Mientras $(i < n)$ **y** $(A[i] \neq K)$ **hacer**

$i \leftarrow i + 1$

Fin Mientras

Si $(i < n)$ **entonces**

retornar i

Sino

retornar -1

Fin Si

Procesamiento de Cadenas

- String: Secuencia de caracteres que pertenecen a un alfabeto
 - Text Strings: letras + números + caracteres especiales

ABCDEFGHIJKLMN
OPQRSTUVWXYZÀ
abcdefghijklmnopqrstuvwxyz
stuvwxyzàâéîõöü&
1234567890(\$£€.,!?)

- Bit Strings: 0's y 1's

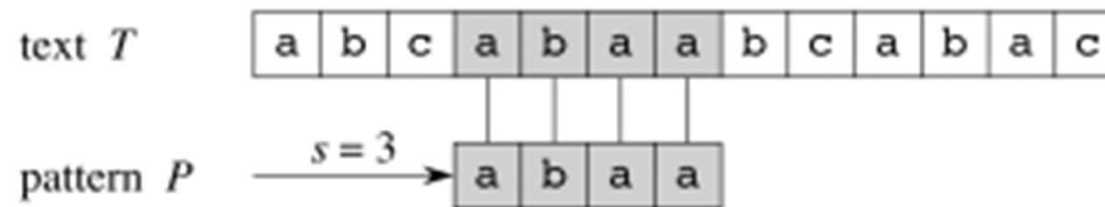
10000010001011111100111111110110110001...

- Gene Sequence: modelado como cadena de caracteres del alfabeto {A, C, G, T}

ATCTCTTGGCTCCAGCATCGATGAAGAACGCA
TCATTTAGAGGAAGTAAAGTCGTAAACAAGGT
GAACGTCAAACTTTTAAACGATCTCTT
TGTTCCTCGCCGGCCGCCAAGGGTGCCCG
GGCCTGCCGTGGCAGATCCCAACGCCGGCC
TCTCTTGGCTCCAGCATCGATGAAGAACGCA
CAGCATCGATGAAGAACGAGCGAAACGCGAT
CGATCTCTTGGCTCCAGCATCGATGAAGAAC
CGGATCTCTTGGCTCCAGCATCGATGAAGAAC
GATGAAGAACGAGCGAAACGCGATCTCTT

String Matching

- Dado una cadena de caracteres de tamaño n llamada texto, y una cadena de caracteres de tamaño m llamada patrón ($m < n$), se busca una subcadena de caracteres dentro del texto que coincida con el patrón



Ejemplo

N O B O D Y _ N O T I C E D → Texto

N O T → Patrón

N O T

N O T

N O T

N O T

N O T

N O T

N O T → Encontrado en posición 8

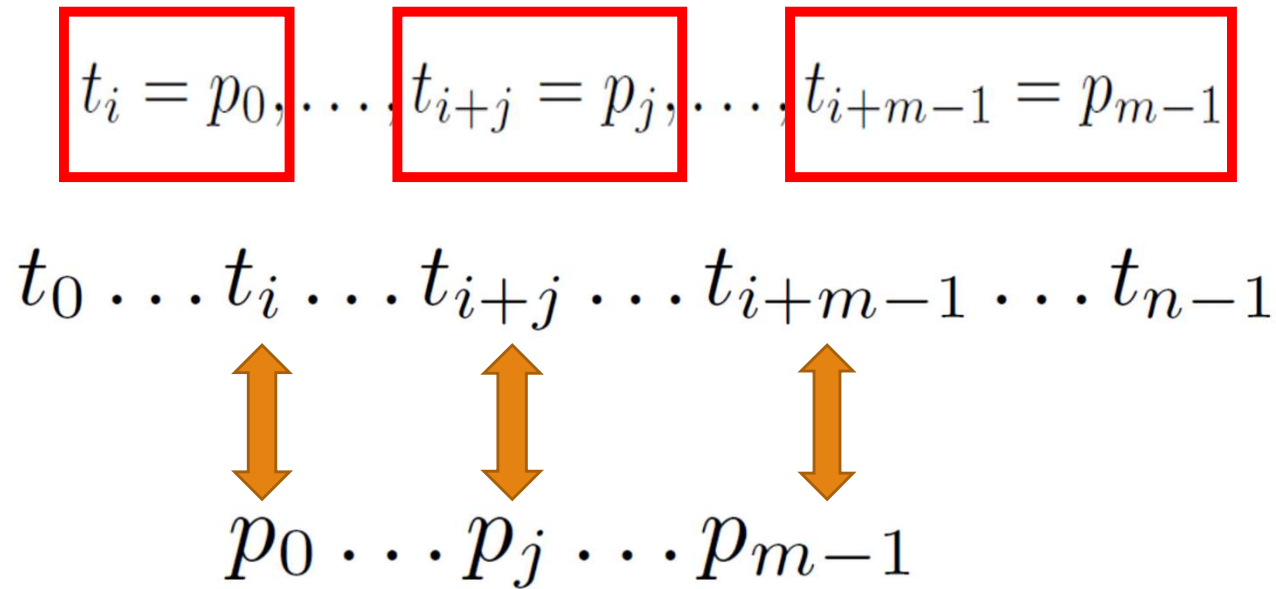
String Matching Estrategia

- Se alinea el patrón con los primeros m caracteres del texto. Se verifica carácter a carácter la ocurrencia del patrón en el texto.
 - Si todos los caracteres del patrón coinciden, el algoritmo termina.
 - Si algún carácter no coincide, el patrón no está en la primera posición del texto.
 - Se desplaza el patrón un carácter a la derecha y se repite el proceso.

String Matching

Estrategia

- Problema se reduce a buscar la posición i del carácter más a la izquierda de la primera ocurrencia del patrón en el texto.



String Matching

Pseudocódigo

Precondición: Arreglo $T[0..n-1]$ de n caracteres que representa el texto, y un arreglo $P[0..m-1]$ de m caracteres que representa el patrón

Postcondición: El índice del primer caracter de T en donde existe una coincidencia del patrón, o -1 si el patrón no se encuentra

```
Para  $i=0$  hasta  $n-m$  hacer
     $j \leftarrow 0$ 
    Mientras  $(j < m)$  y  $(P[j] = T[i+j])$  hacer
         $j \leftarrow j+1$ 
    Fin Mientras
    Si  $j=m$  entonces
        retornar  $i$ 
    Fin Si
Fin Para
retornar  $-1$ 
```

Closest Pair

- Problema: Hallar el par de puntos más cercanos en un conjunto de n puntos.
- Asumimos que:
 - Puntos están en dos dimensiones
 - Puntos especificados por coordenadas cartesianas (x, y)
 - La distancia entre dos puntos $p_i(x_i, y_i)$ y $p_j(x_j, y_j)$ se calcula como

$$d(p_i, p_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}.$$

Closest Pair

- Calcular la distancia entre cada par de puntos y escoger el par con la menor distancia.
- Como la distancia es simétrica, no calculamos la distancia entre un par de puntos más de una vez
 - Se consideran sólo pares de puntos (p_i, p_j) en donde i sea menor que j .

Closest Pair

Estrategia

Precondición: Un conjunto P de n puntos ($n \geq 2$):

$P_1 = (x_1, y_1), \dots, P_n = (x_n, y_n)$

Postcondición: Los índices i_1 e i_2 del par más cercanos

```
dmenor  $\leftarrow$  INF
Para  $i=1$  hasta  $n-1$  hacer
    Para  $j=i+1$  hasta  $n$  hacer
         $d \leftarrow (x_i - x_j)^2 + (y_i - y_j)^2$ 
        Si  $d < dmenor$  entonces
             $dmenor \leftarrow d$ 
             $i_1 \leftarrow i$ 
             $i_2 \leftarrow j$ 
        Fin Si
    Fin Para
Fin Para
retornar  $i_1, i_2$ 
```

Análisis de Eficiencia

Ordenación por Selección

Pseudocódigo

Precondición: Arreglo $A[0..n-1]$ desordenado

Postcondición: Arreglo $A[0..n-1]$ ordenado

```
Para  $i \leftarrow 0$  hasta  $n-2$  hacer  $\leftarrow \sum_{i=0}^{n-2}$ 
     $\text{min} \leftarrow i$ 
    Para  $j \leftarrow i+1$  hasta  $n-1$  hacer  $\leftarrow \sum_{j=i+1}^{n-1}$ 
        Si  $A[j] < A[\text{min}]$  entonces
             $\text{min} \leftarrow j$ 
        Fin Si
    Fin Para
    intercambiar  $A[i]$  y  $A[\text{min}]$ 
Fin Para
```

$O(1)$

Ordenación por Selección

Eficiencia

$$\begin{aligned}C(n) &= \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1 \\&= \sum_{i=0}^{n-2} [(n-1) - (i+1) + 1] \\&= \sum_{i=0}^{n-2} n - 1 - i = \frac{n \times (n-1)}{2} \\&O(n^2)\end{aligned}$$

Ordenación Burbuja

Pseudocódigo

Precondición: Arreglo $A[0..n-1]$ desordenado

Postcondición: Arreglo $A[0..n-1]$ ordenado

Para $i \leftarrow 0$ **hasta** $n-2$ **hacer**

Para $j \leftarrow 0$ **hasta** $n-2-i$ **hacer**

Si $A[j+1] < A[j]$ **entonces**

intercambiar $A[j]$ y $A[j+1]$

Fin Si

Fin Para

Fin Para

Ordenación Burbuja

Eficiencia

$$\begin{aligned}C(n) &= \sum_{i=0}^{n-2} \sum_{j=0}^{n-2-i} 1 \\&= \sum_{i=0}^{n-2} n - 2 - i + 1 = \sum_{i=0}^{n-2} n - i - 1 \\&= \sum_{i=0}^{n-2} n - 1 - i = \frac{n \times (n - 1)}{2} \\&\quad O(n^2)\end{aligned}$$

Búsqueda Secuencial

Pseudocódigo

Precondición: Arreglo $A[0..n-1]$ y una llave de búsqueda K

Postcondición: El índice del primer elemento de A que coincide con la llave K o -1 si la llave no se encuentra

$i \leftarrow 0$

Mientras $(i < n)$ **y** $(A[i] \neq K)$ **hacer**

$i \leftarrow i + 1$

Fin Mientras

Si $(i < n)$ **entonces**

retornar i

Sino

retornar -1

Fin Si

$$C(n)_{peor} = n$$

String Matching

Pseudocódigo

Precondición: Arreglo $T[0..n-1]$ de n caracteres que representa el texto, y un arreglo $P[0..m-1]$ de m caracteres que representa el patrón

Postcondición: El índice del primer caracter de T en donde existe una coincidencia del patrón, o -1 si el patrón no se encuentra

Para $i=0$ **hasta** $n-m$ **hacer**

$j \leftarrow 0$

Mientras $(j < m)$ **y** $(P[j] = T[i+j])$ **hacer**

$j \leftarrow j+1$

Fin Mientras

Si $j=m$ **entonces**

 retornar i

Fin Si

Fin Para

retornar -1

$$C(n, m)_{peor} = (n - m + 1) \times m$$

$$O(n \times m)$$

Closest Pair

Estrategia

Precondición: Un conjunto P de n puntos ($n \geq 2$):

$P_1 = (x_1, y_1), \dots, P_n = (x_n, y_n)$

Postcondición: Los índices i_1 e i_2 del par más cercanos

```
dmenor  $\leftarrow$  INF
Para  $i=1$  hasta  $n-1$  hacer
    Para  $j=i+1$  hasta  $n$  hacer
         $d \leftarrow (x_i - x_j)^2 + (y_i - y_j)^2$ 
        Si  $d < dmenor$  entonces
             $dmenor \leftarrow d$ 
             $i_1 \leftarrow i$ 
             $i_2 \leftarrow j$ 
        Fin Si
    Fin Para
Fin Para
retornar  $i_1, i_2$ 
```

Closest Pair

Eficiencia

$$\begin{aligned}C(n) &= \sum_{i=1}^{n-1} \sum_{j=i+1}^n 2 \\&= 2 \times \sum_{i=1}^{n-1} (n - i) \\&= 2 \times [(n - 1) + (n - 2) + \dots + 1] = n \times (n - 1)\end{aligned}$$

$$O(n^2)$$

Resumen

- Estrategia útil cuando el problema tiene pocos datos.
- Fácil de aplicar en casi todos los problemas, no necesariamente con las soluciones más eficientes.
- Diseñar un mejor algoritmo sólo vale la pena si los problemas a resolver tienen instancias con muchos datos.
- Ventajas: Aplicabilidad y simplicidad
- Desventaja: Ineficiencia

Resumen

Ventajas

- Amplia aplicabilidad
- Simplicidad
- Permite resolver problemas importantes como ordenamientos, manejo de cadenas, búsquedas, matrices, etc.
- Se pueden utilizar como base para desarrollar algoritmos mas eficientes

Desventaja

- No son eficientes (ineficiencia)
- Son extremadamente lentos, se incrementa con mayor cantidad de datos

Bibliografía recomendada

- LEVITIN, A. **Introduction to The Design and Analysis of Algorithms**. 3ra edición. USA: Pearson, 2012. ISBN 0-13-231681-1.
 - Cap3: Brute Force and Exhaustive Search
 - 3.1 Selection Sort and Bubble Sort
 - 3.2 Sequential Search and Brute-Force String Matching
 - 3.3 Closest Pair [...] Problems by Brute Force
- Diapositivas basadas en dispositivas del Prof. Fernando Alva.