


El problema del ordenamiento

Algoritmos de ordenamiento

- **Entrada:** Una secuencia de n números $\{a_1, a_2, \dots, a_n\}$
- **Salida:** Una permutación (reordenamiento) $\{a'_1, a'_2, \dots, a'_n\}$ de la secuencia de entrada tal que:
$$a'_1 \leq a'_2 \leq \dots \leq a'_n$$
- Dada una subsecuencia de entrada como:
 $\{31, 41, 59, 26, 41, 58\}$
- Un algoritmo de ordenamiento retorna como salida la secuencia:
 $\{26, 31, 41, 41, 58, 59\}$

- 
- Qué algoritmo de ordenamiento es el mejor para una aplicación dada depende de:
 - El número de elementos a ser ordenados.
 - La medida en que los elementos ya están algo ordenados.
 - El algoritmo de ordenamiento debe de proporcionar una descripción precisa del procedimiento computacional a ser seguido.

Ordenamiento por inserción

Ordenamiento por inserción

- Algoritmo eficiente para ordenar una pequeña cantidad de elementos.
- La ordenación por inserción funciona de la misma manera en que muchas personas ordenan una mano de cartas.



Ordenamiento por inserción

El ordenamiento por inserción analiza de izquierda a derecha, uno por uno, los datos de una sucesión. Una vez que se ha analizado cada dato, se inserta en un sitio idóneo en una sucesión ya ordenada. Por ejemplo, suponga que la sucesión de datos a ordenar es

11, 7, 14, 1, 5, 9, 10.

El ordenamiento por inserción funciona sobre la sucesión anterior como se muestra a continuación:

Sucesión ordenada

11

7, 11

7, 11, 14

1, 7, 11, 14

1, 5, 7, 11, 14

1, 5, 7, 9, 11, 14

1, 5, 7, 9, 10, 11, 14

Sucesión no ordenada

7, 14, 1, 5, 9, 10

14, 1, 5, 9, 10

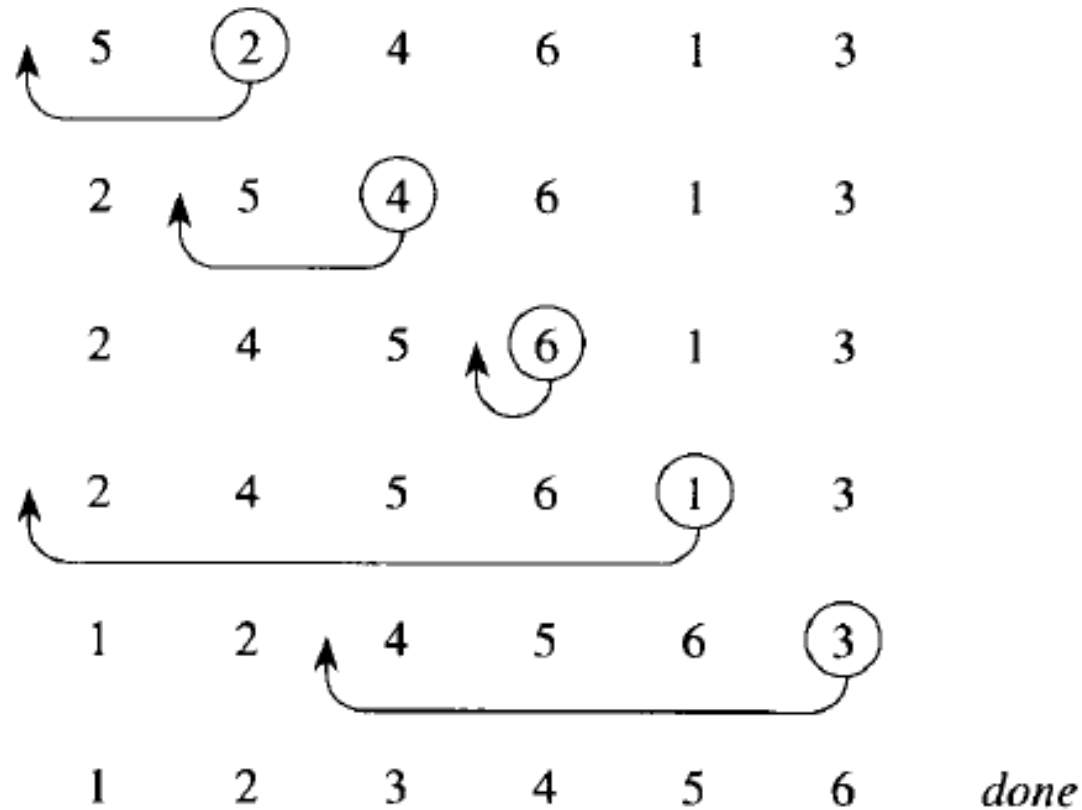
1, 5, 9, 10

5, 9, 10

9, 10

10

Ordenamiento por inserción



Ordenamiento por inserción

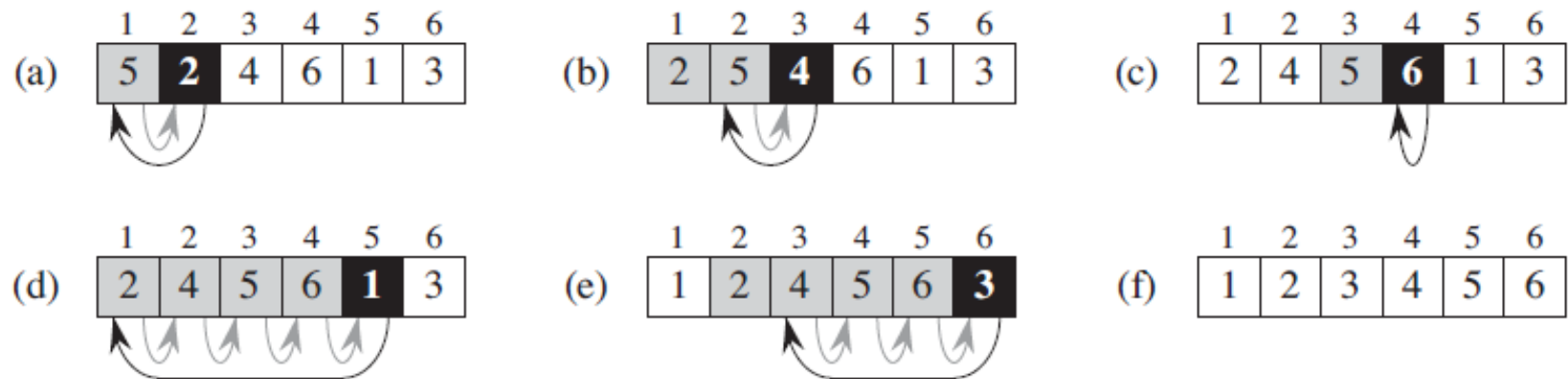


Figure 2.2 The operation of INSERTION-SORT on the array $A = \langle 5, 2, 4, 6, 1, 3 \rangle$. Array indices appear above the rectangles, and values stored in the array positions appear within the rectangles. (a)–(e) The iterations of the **for** loop of lines 1–8. In each iteration, the black rectangle holds the key taken from $A[j]$, which is compared with the values in shaded rectangles to its left in the test of line 5. Shaded arrows show array values moved one position to the right in line 6, and black arrows indicate where the key moves to in line 8. (f) The final sorted array.

Ordenamiento por inserción

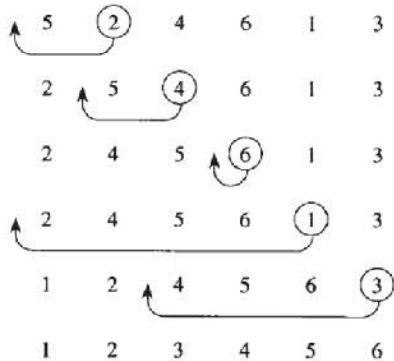
INSERTION-SORT(A)

```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3      // Insert  $A[j]$  into the sorted sequence  $A[1..j-1]$ .
4       $i = j - 1$ 
5      while  $i > 0$  and  $A[i] > key$ 
6           $A[i + 1] = A[i]$ 
7           $i = i - 1$ 
8       $A[i + 1] = key$ 
```

Prueba de escritorio

- Una **prueba de escritorio** es un tipo de prueba algorítmica, que consiste en la validación y verificación del algoritmo a través de la ejecución de las sentencias que lo componen (proceso) para determinar sus resultados (salida) a partir de un conjunto determinado de elementos (entrada).

Prueba de escritorio



INSERTION-SORT(A)

```

1 for  $j \leftarrow 2$  to  $\text{length}[A]$ 
2   do  $\text{key} \leftarrow A[j]$ 
3      $\triangleright$  Insert  $A[j]$  into the sorted sequence  $A[1..j-1]$ .
4      $i \leftarrow j-1$ 
5     while  $i > 0$  and  $A[i] > \text{key}$ 
6       do  $A[i+1] \leftarrow A[i]$ 
7          $i \leftarrow i-1$ 
8      $A[i+1] \leftarrow \text{key}$ 

```

→ $j = 6$
key = 3
 $i = 5$

$5 > 0 \ \&\& \ 6 > 3$
 $A[6] = 6$
 $i = 4$

$4 > 0 \ \&\& \ 6 > 1$
 $A[5] = 6$
 $i = 3$

$3 > 0 \ \&\& \ 5 > 1$
 $A[4] = 5$
 $i = 2$

$2 > 0 \ \&\& \ 4 > 1$
 $A[3] = 4$
 $i = 1$

$1 > 0 \ \&\& \ 2 > 1$
 $A[2] = 2$
 $i = 0$

$A[1] = 1$

$4 > 0 \ \&\& \ 5 > 3$
 $A[5] = 5$
 $i = 3$

$3 > 0 \ \&\& \ 4 > 3$
 $A[4] = 4$
 $i = 2$

$2 > 0 \ \&\& \ 2 > 3$
 $A[3] = 3$

length[A] = 6
→ $j = 2$
key = 2
 $i = 1$

$1 > 0 \ \&\& \ 5 > 2$
 $A[2] = 5$
 $i = 0$

$A[1] = 2$

→ $j = 3$
key = 4
 $i = 2$

$2 > 0 \ \&\& \ 5 > 4$
 $A[3] = 5$
 $i = 1$

$1 > 0 \ \&\& \ 2 > 4$

$A[2] = 4$

→ $j = 4$
key = 6
 $i = 3$

$3 > 0 \ \&\& \ 5 > 6$

$A[4] = 6$

→ $j = 5$
key = 1

$i = 4$

Complejidad algorítmica del algoritmo de inserción

$$O(n^2)$$