Estructuras de datos

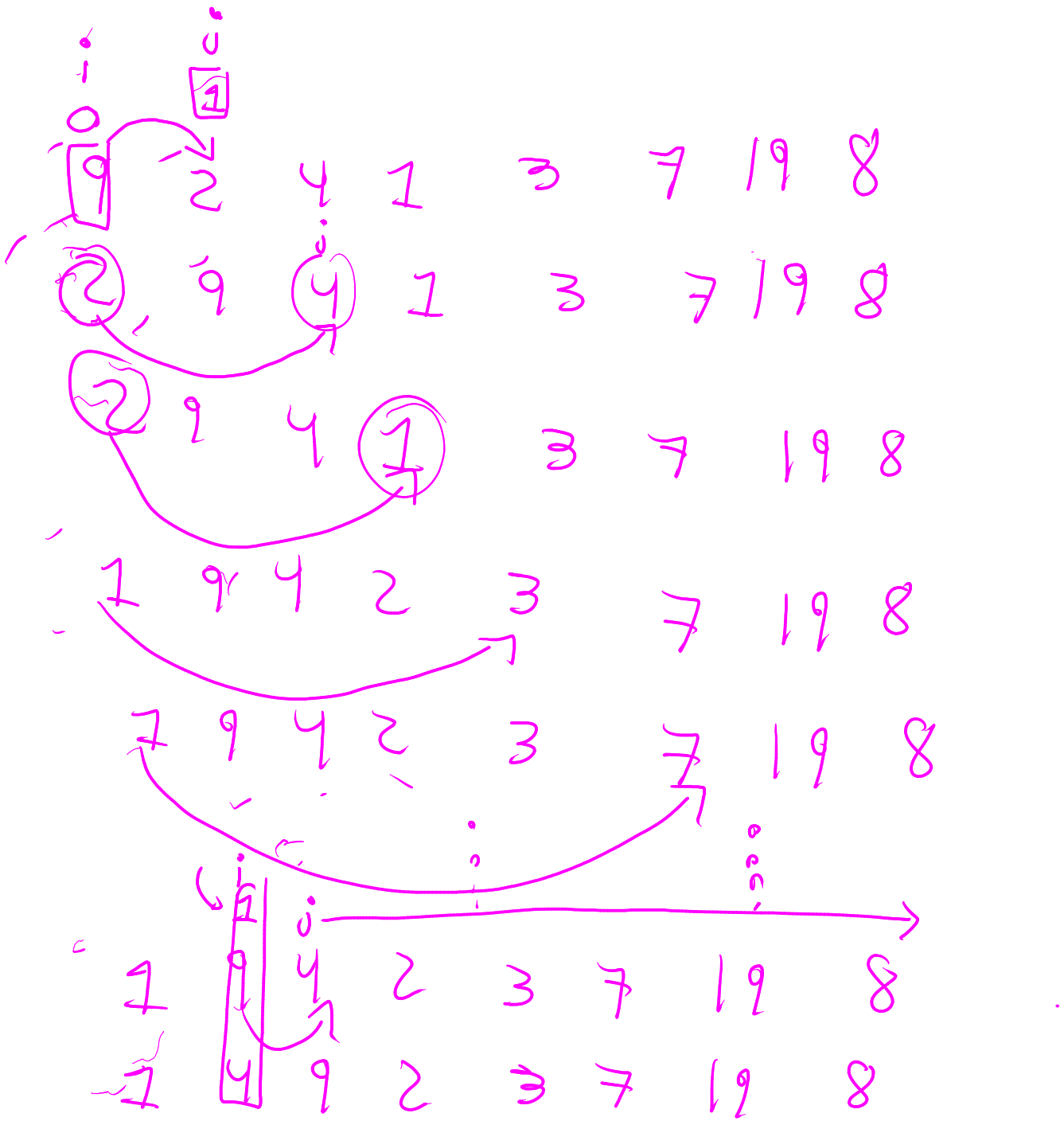
Algoritmos insertionSort

1 4 2 3 7 8 0

0 1 2 3 4 7 8

In place = In situ >> modifican estructure

No son in place = No retorna uno nuevo



Algoritmos de Ordenamiento

Analicemos un ejemplo (Algoritmo de ordenamiento insertion-sort)

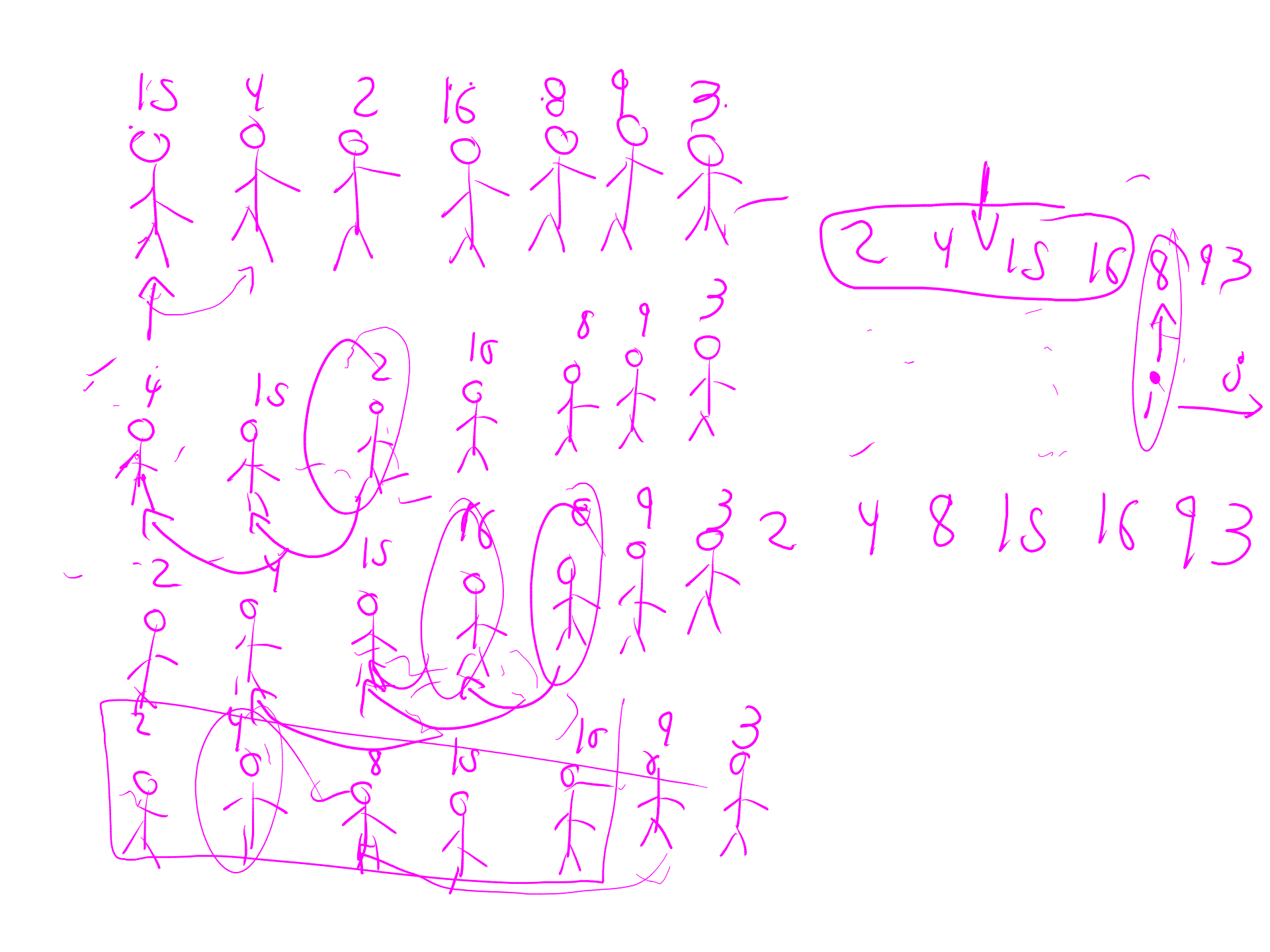
	Instrucción	Costo	Veces que se repite
1	for j←2 to length[A]		
2	do key←A[j]		
3	i←j-1		
4	while i >0 and A[i] > key		
5	do A[i+1]←A[i]		
6	i←i-1		
7	A[i+1]←key		

¡Sin temor!, vamos a explorar este algoritmo.

Windows (CMD, power shell)
Instalado el Python con la opción de PATH
py -m pip install matplotlib
py -m pip install numpy

Linux
Debian/Ubuntu
sudo apt-get install python3
sudo apt-get install python3-pip
sudo pip3 install matplotlib
sudo pip3 install numpy

MAC
brew install python3


15 4 2 16 8 9 3
2 4 15 16 8 9 3
i
j
4 15 2 16 8 9 3
2 4 8 15 16 9 3
2 4 15 16 8 9 3
2 4 8 15 16 9 3

Algoritmos de Ordenamiento

Este algoritmo recibe un arreglo de tamaño n y retorna el mismo arreglo ordenado de menor a mayor.

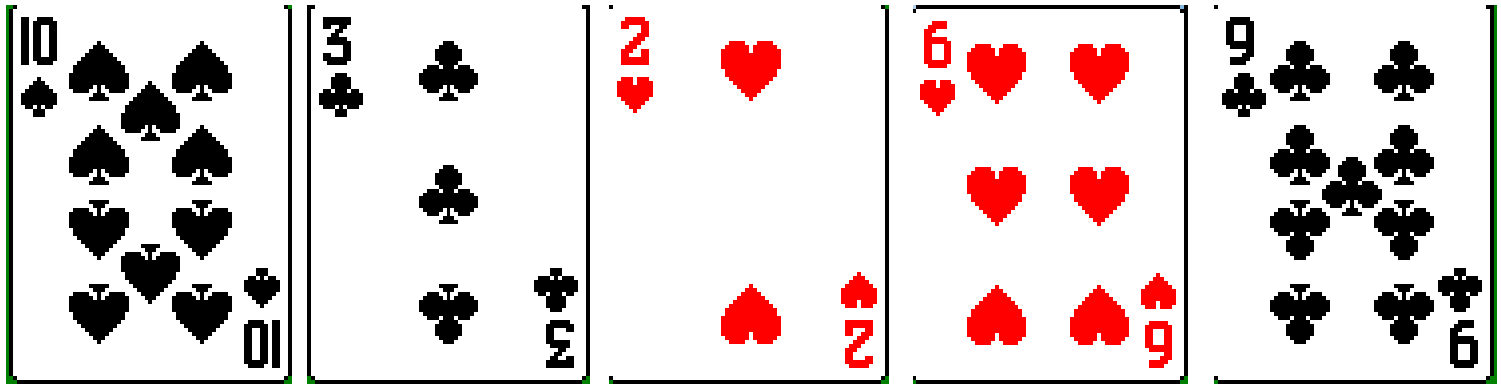
Ejemplo

Entrada = {10,3,2,6,9}

Salida = {2,3,6,9,10}

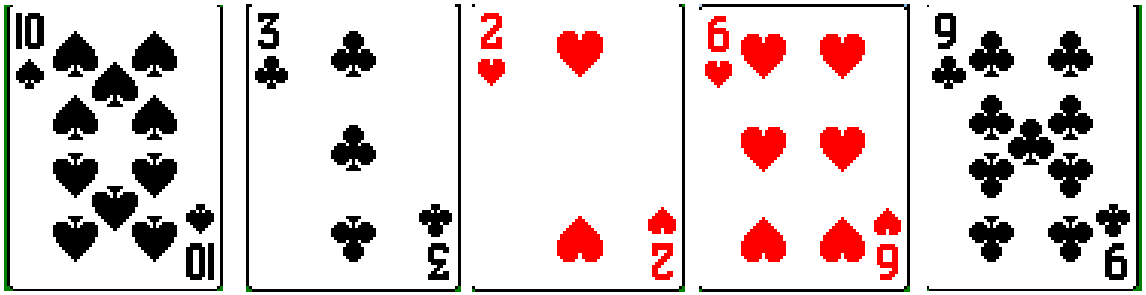
Algoritmos de Ordenamiento

Insertion sort



Algoritmos de Ordenamiento

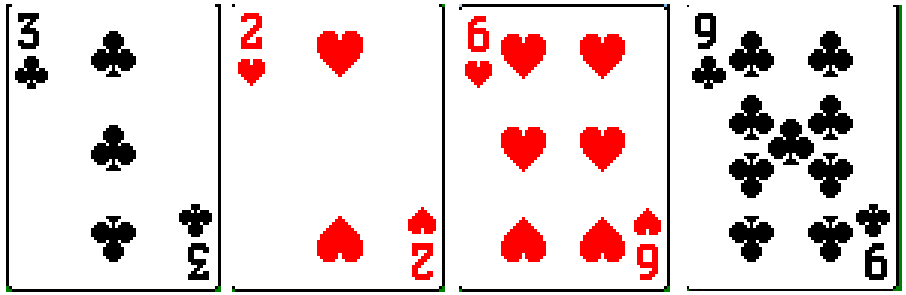
Insertion sort

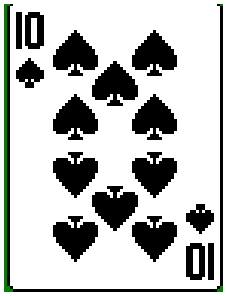


```
1  for j←2 to length[A]
2        do key←A[j]
3           i←j-1
4           while i >0 and A[i] > key
5                 do A[i+1]←A[i]
6                    i←i-1
7           A[i+1]←key
```

Algoritmos de Ordenamiento

Insertion sort

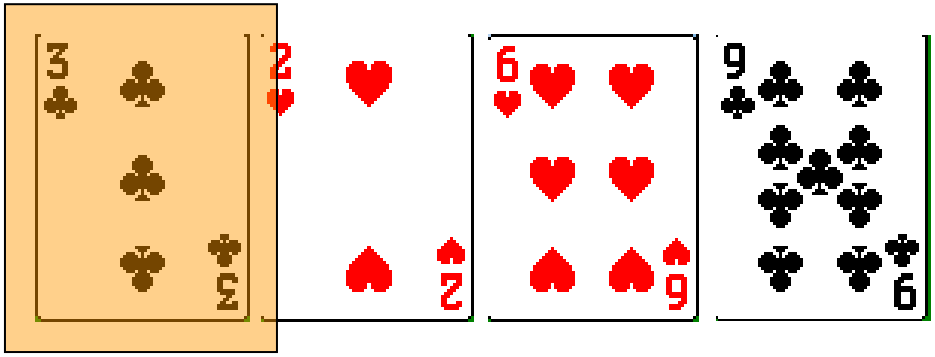


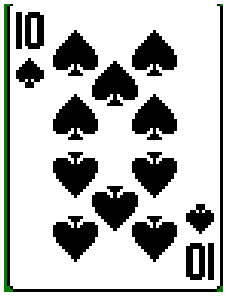


```
1  for j←2 to length[A]
2        do key←A[j]
3           i←j-1
4           while i >0 and A[i] > key
5                 do A[i+1]←A[i]
6                    i←i-1
7           A[i+1]←key
```

Algoritmos de Ordenamiento

Insertion sort

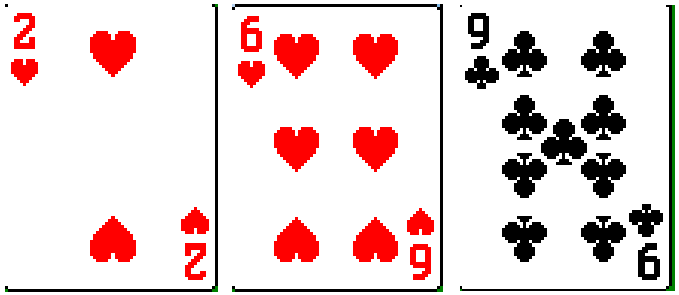


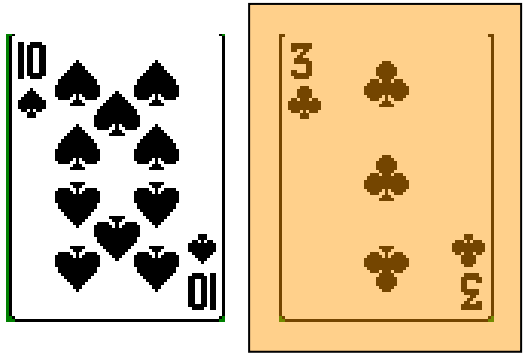


1	for j←2 to length[A]
2	do key←A[j]
3	i←j-1
4	while i >0 and A[i] > key
5	do A[i+1]←A[i]
6	i←i-1
7	A[i+1]←key

Algoritmos de Ordenamiento

Insertion sort



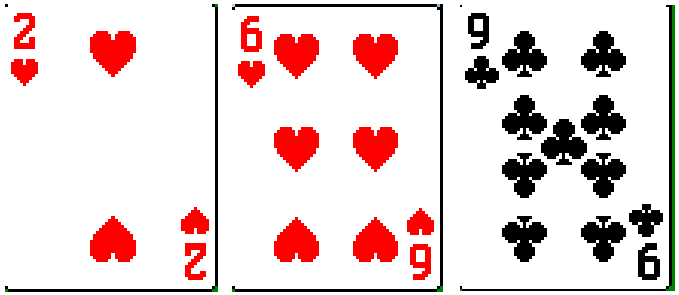


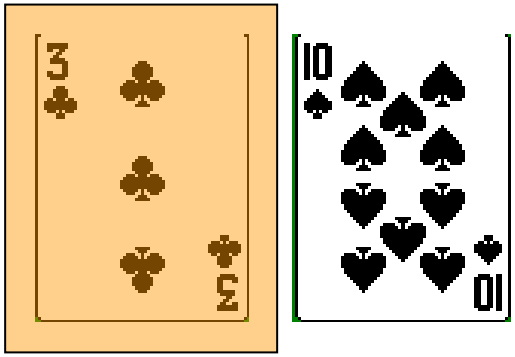
Se recorre de derecha a izquierda buscando el lugar que debe ocupar

```
1  for j←2 to length[A]
2        do key←A[j]
3           i←j-1
4           while i >0 and A[i] > key
5                 do A[i+1]←A[i]
6                    i←i-1
7           A[i+1]←key
```

Algoritmos de Ordenamiento

Insertion sort

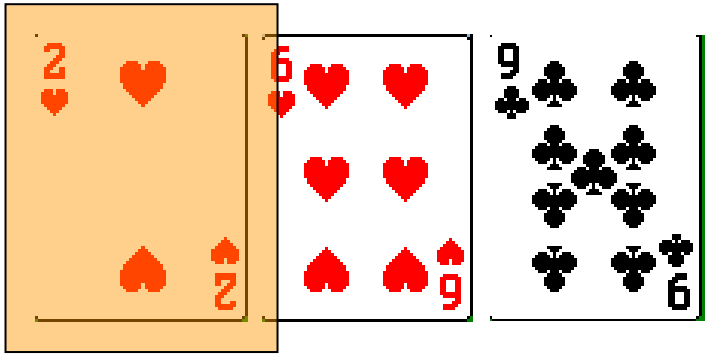


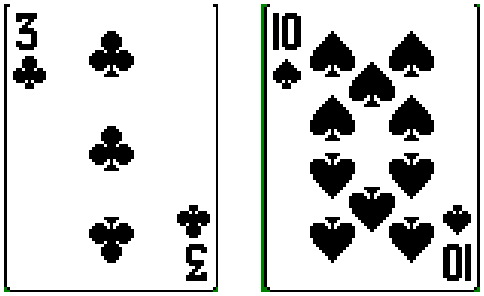


1	for j←2 to length[A]
2	do key←A[j]
3	i←j-1
4	while i >0 and A[i] > key
5	do A[i+1]←A[i]
6	i←i-1
7	A[i+1]←key

Algoritmos en la computación

Insertion sort

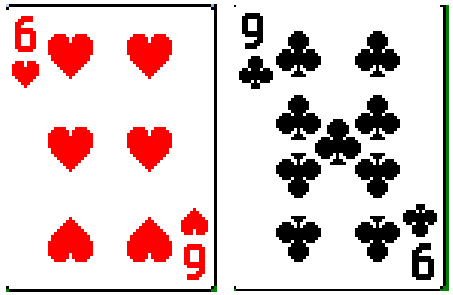


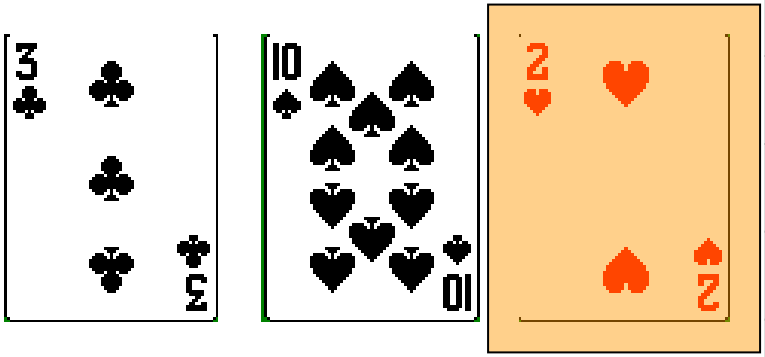


```
1  for j←2 to length[A]
2        do key←A[j]
3           i←j-1
4           while i >0 and A[i] > key
5                 do A[i+1]←A[i]
6                    i←i-1
7           A[i+1]←key
```

Algoritmos en la computación

Insertion sort

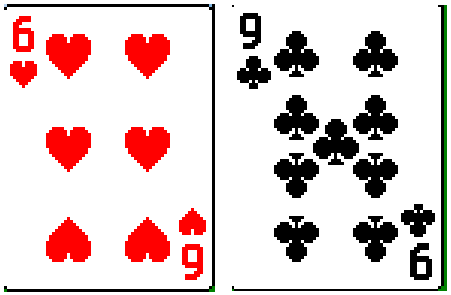


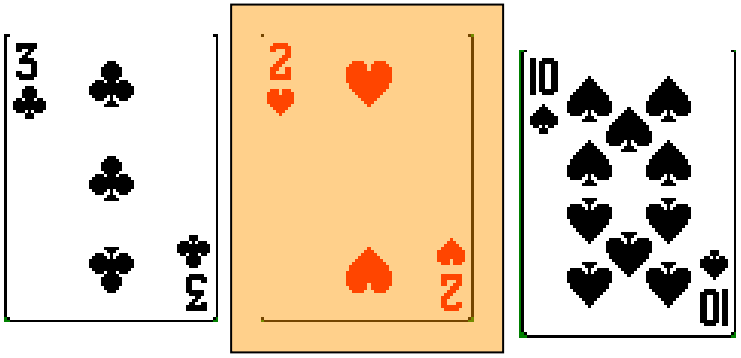


```
1  for j←2 to length[A]
2        do key←A[j]
3           i←j-1
4           while i >0 and A[i] > key
5                 do A[i+1]←A[i]
6                    i←i-1
7           A[i+1]←key
```

Algoritmos en la computación

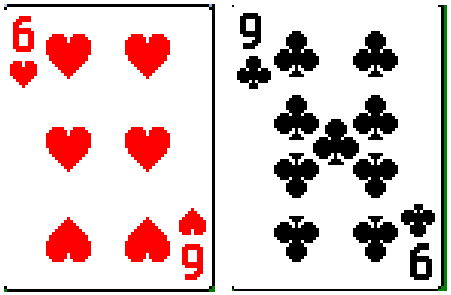
Insertion sort

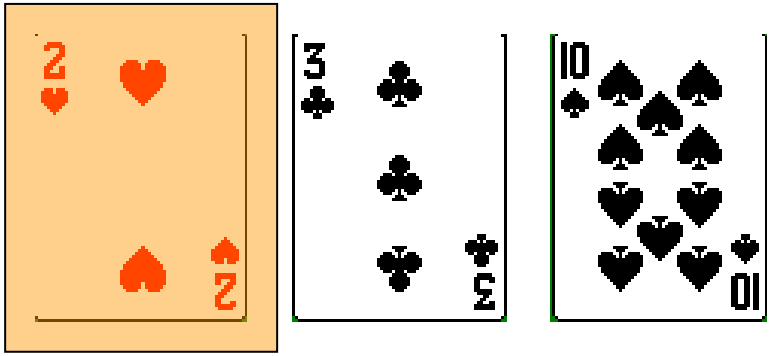




Algoritmos en la computación

Insertion sort

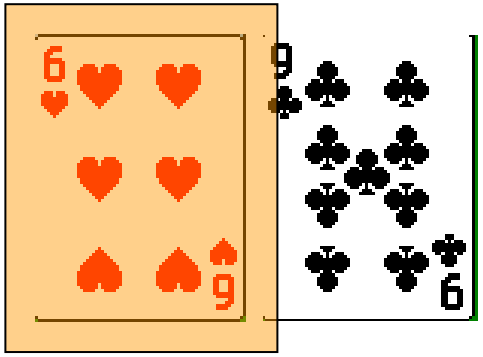


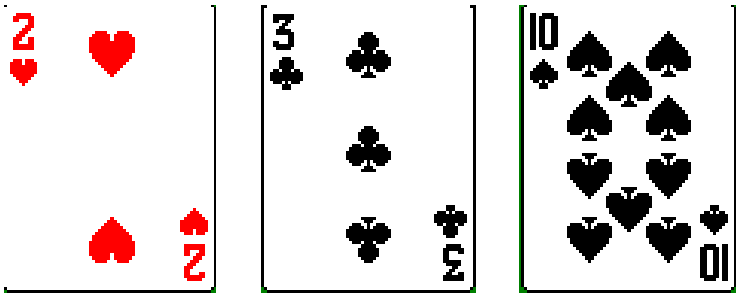


```
1  for j←2 to length[A]
2        do key←A[j]
3           i←j-1
4           while i >0 and A[i] > key
5                 do A[i+1]←A[i]
6                    i←i-1
7           A[i+1]←key
```

Algoritmos de Ordenamiento

Insertion sort

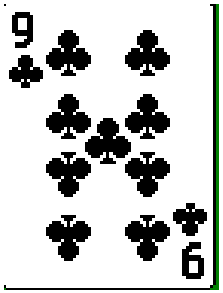


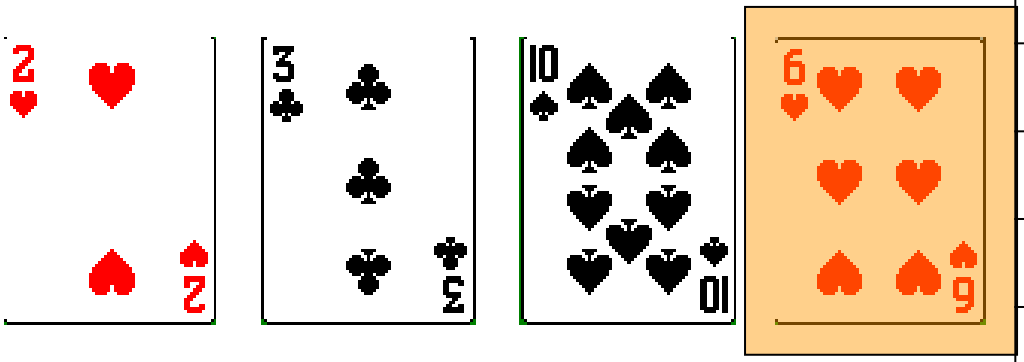


```
1  for j←2 to length[A]
2        do key←A[j]
3           i←j-1
4           while i >0 and A[i] > key
5                 do A[i+1]←A[i]
6                    i←i-1
7           A[i+1]←key
```

Algoritmos de Ordenamiento

Insertion sort

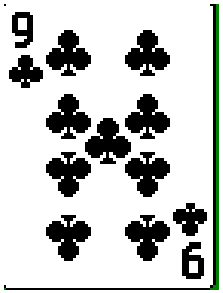


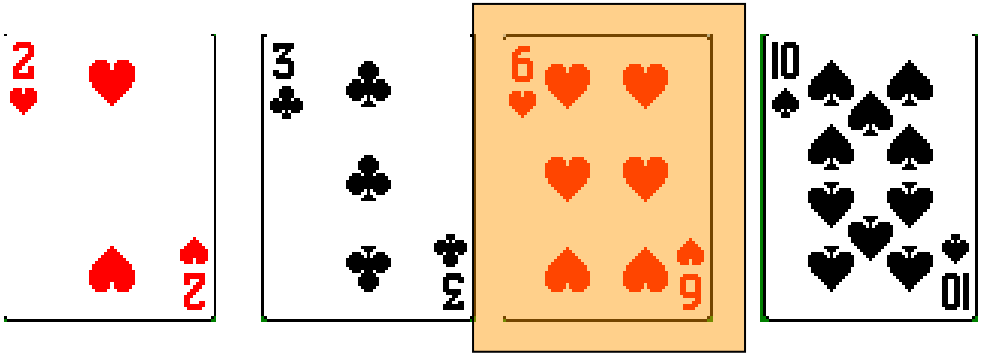


```
1  for j←2 to length[A]
2        do key←A[j]
3           i←j-1
4           while i >0 and A[i] > key
5                 do A[i+1]←A[i]
6                    i←i-1
7           A[i+1]←key
```

Algoritmos de Ordenamiento

Insertion sort

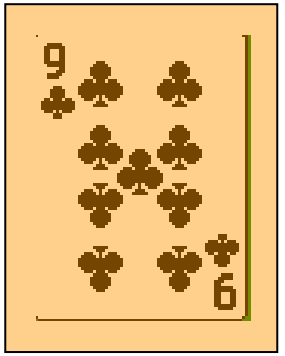


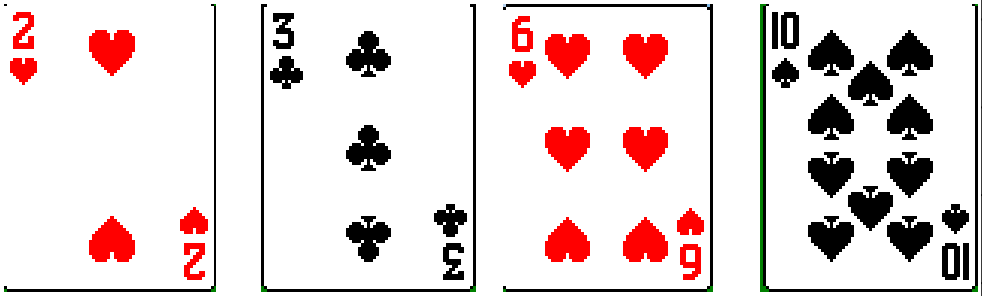


```
1  for j←2 to length[A]
2        do key←A[j]
3           i←j-1
4           while i >0 and A[i] > key
5                 do A[i+1]←A[i]
6                    i←i-1
7           A[i+1]←key
```

Algoritmos de Ordenamiento

Insertion sort

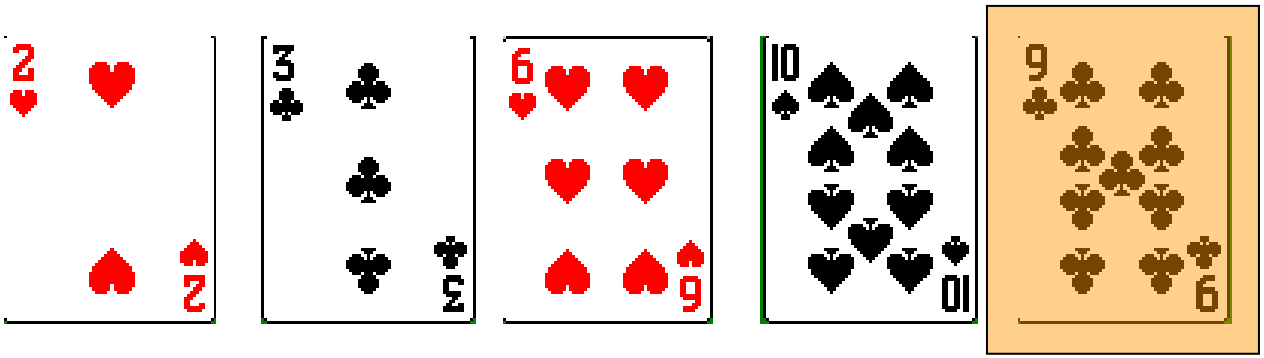




```
1  for j←2 to length[A]
2        do key←A[j]
3           i←j-1
4           while i >0 and A[i] > key
5                 do A[i+1]←A[i]
6                    i←i-1
7           A[i+1]←key
```

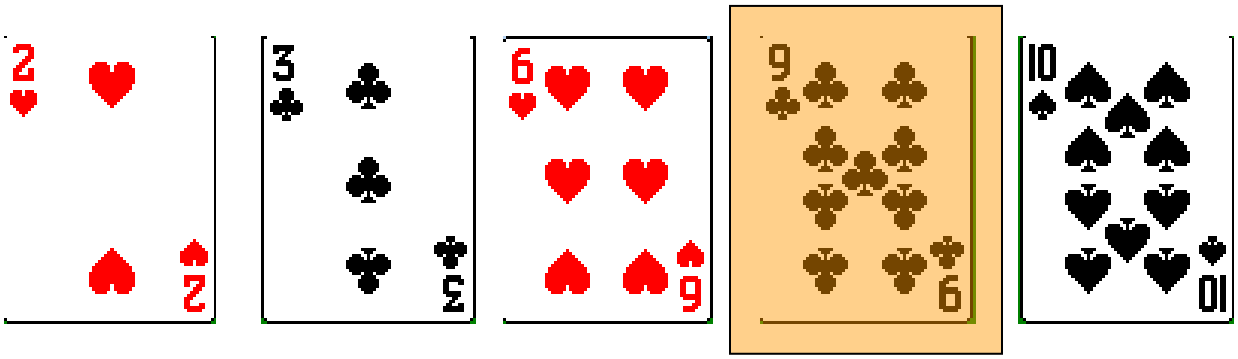
Algoritmos de Ordenamiento

Insertion sort



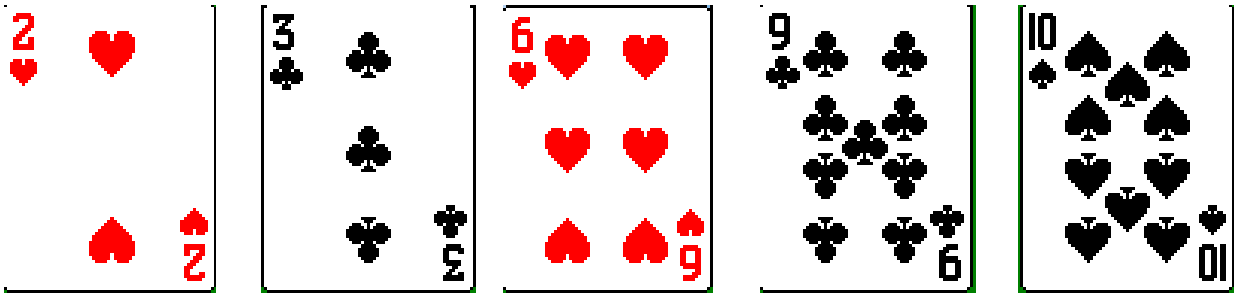
Algoritmos de Ordenamiento

Insertion sort



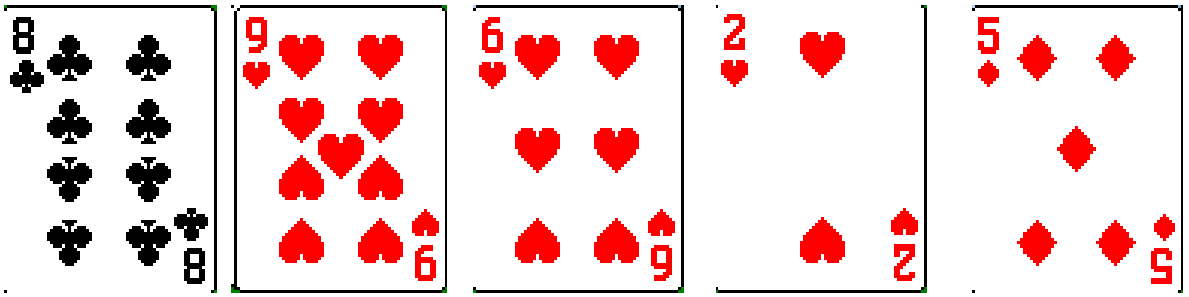
Algoritmos de Ordenamiento

Insertion sort



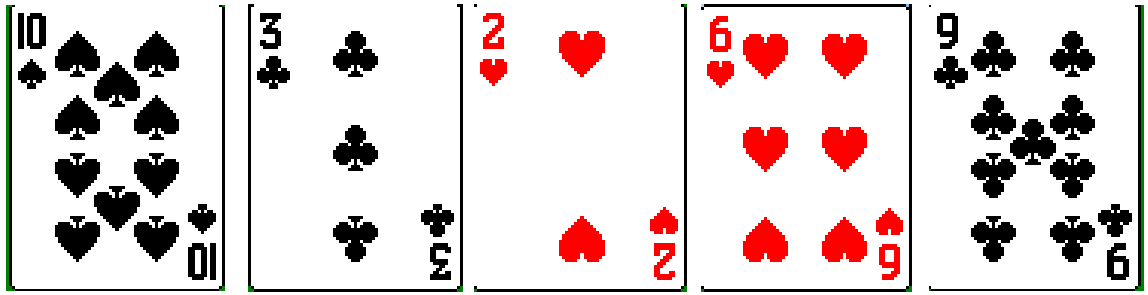
Algoritmos de Ordenamiento

Insertion sort



Algoritmos de Ordenamiento

Insertion sort



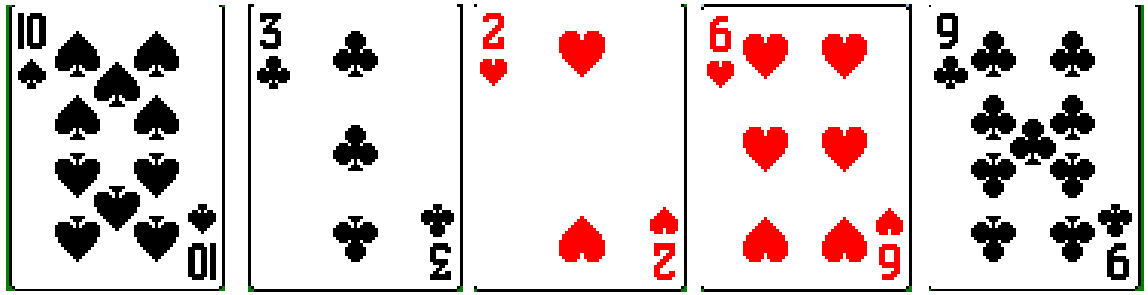
A:	10	3	2	6	9

Algoritmos de Ordenamiento

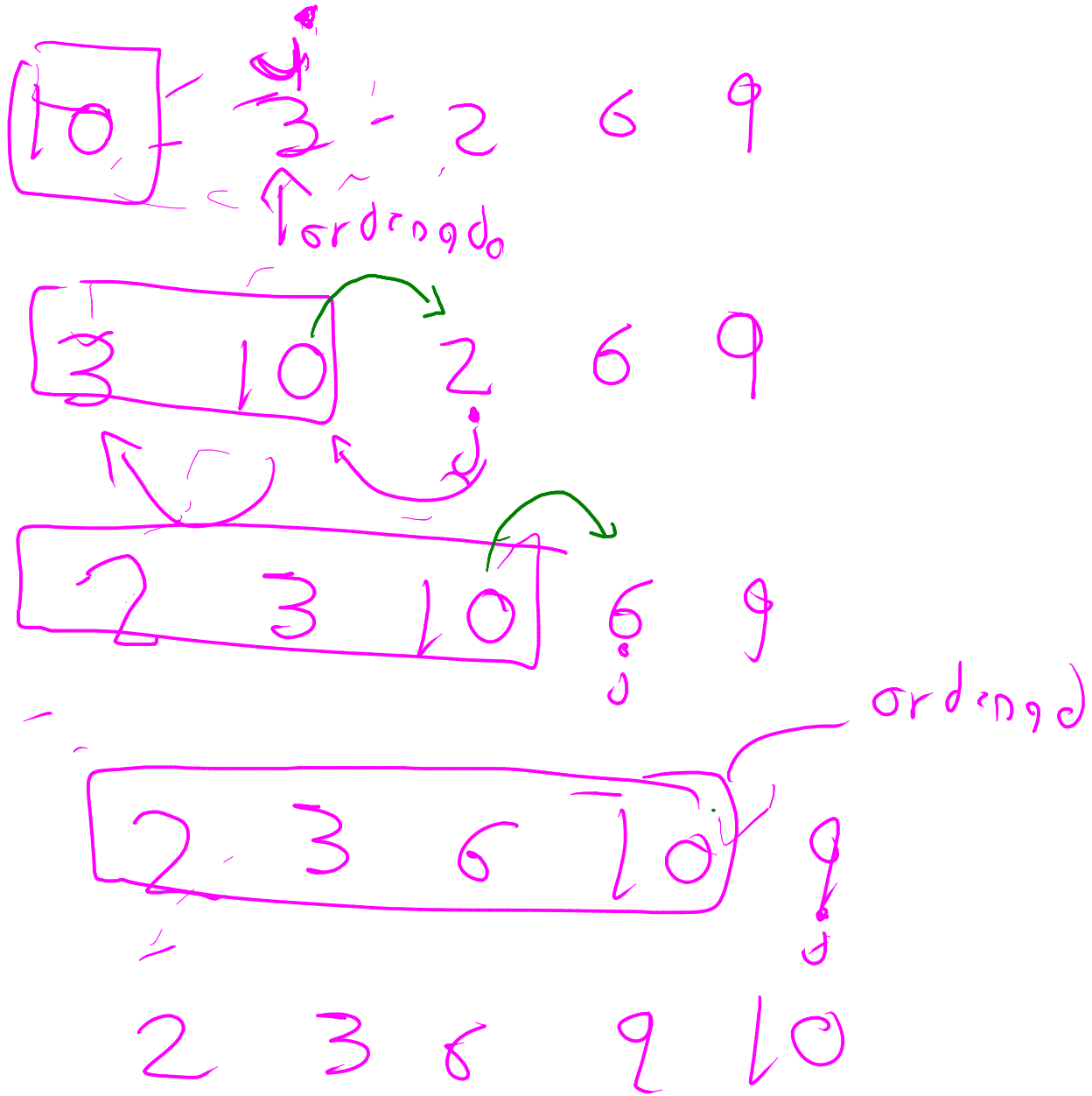
Insertion sort

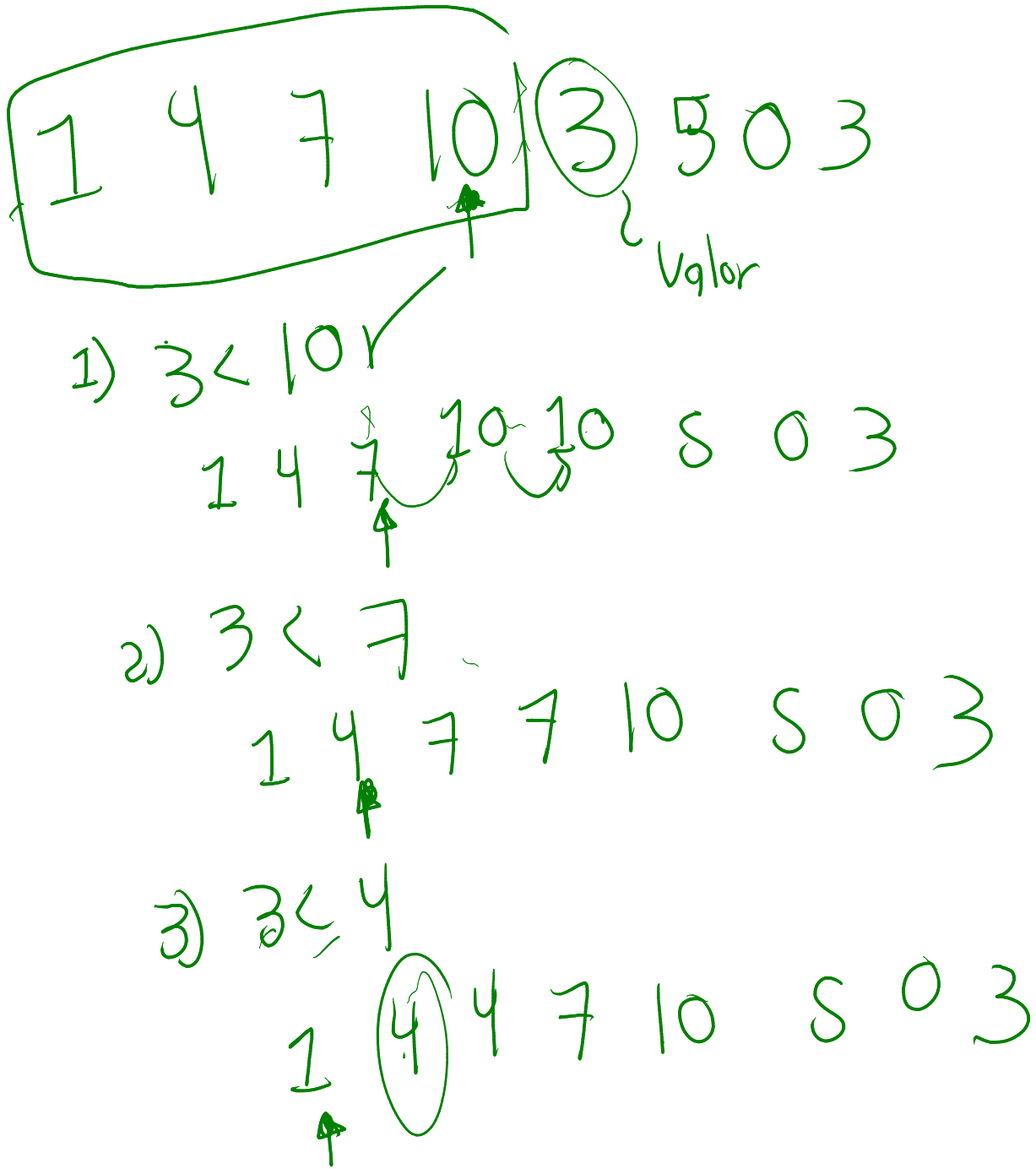
Desarrolle el algoritmo

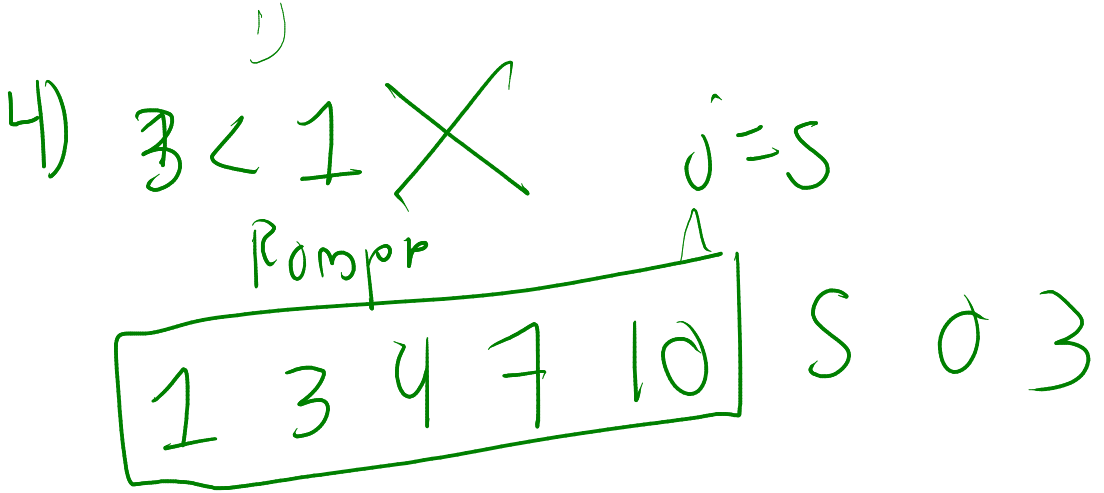
INSERTION-SORT(A)



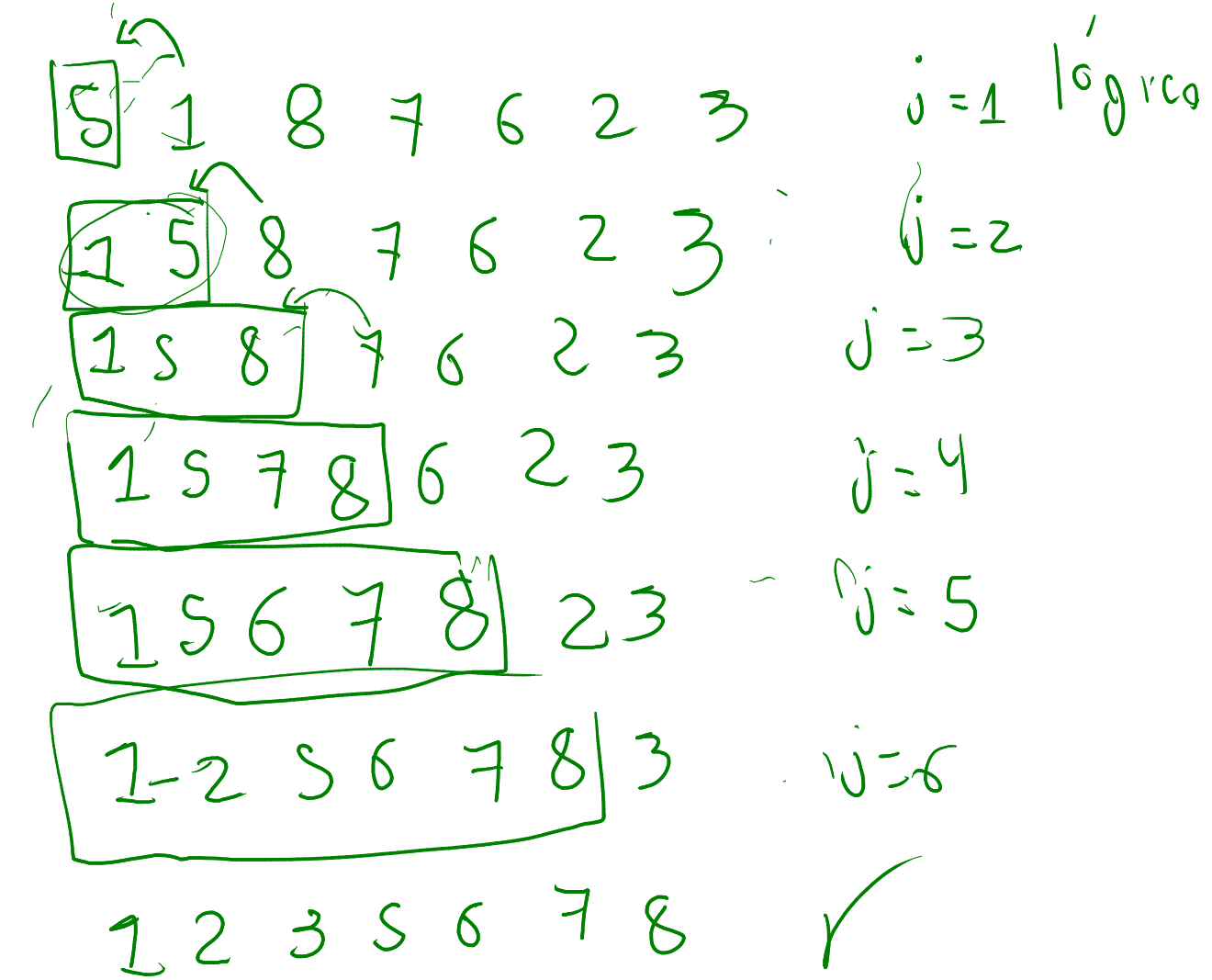
A:	10	3	2	6	9

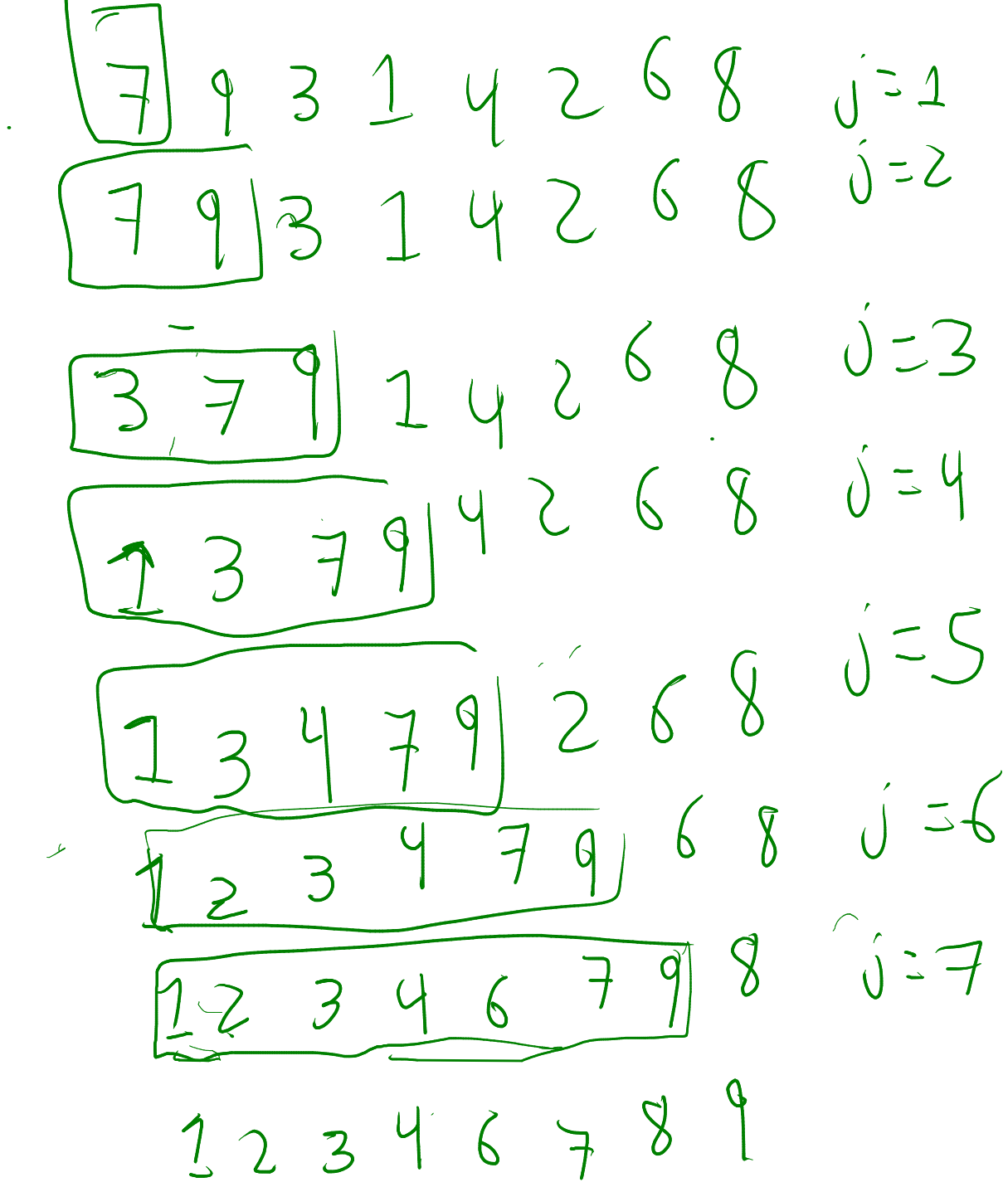

10
3
2
6
9
ordered
3
10
2
6
9
2
3
10
6
9
ordered
2
3
6
10
9
2
3
6
9
10

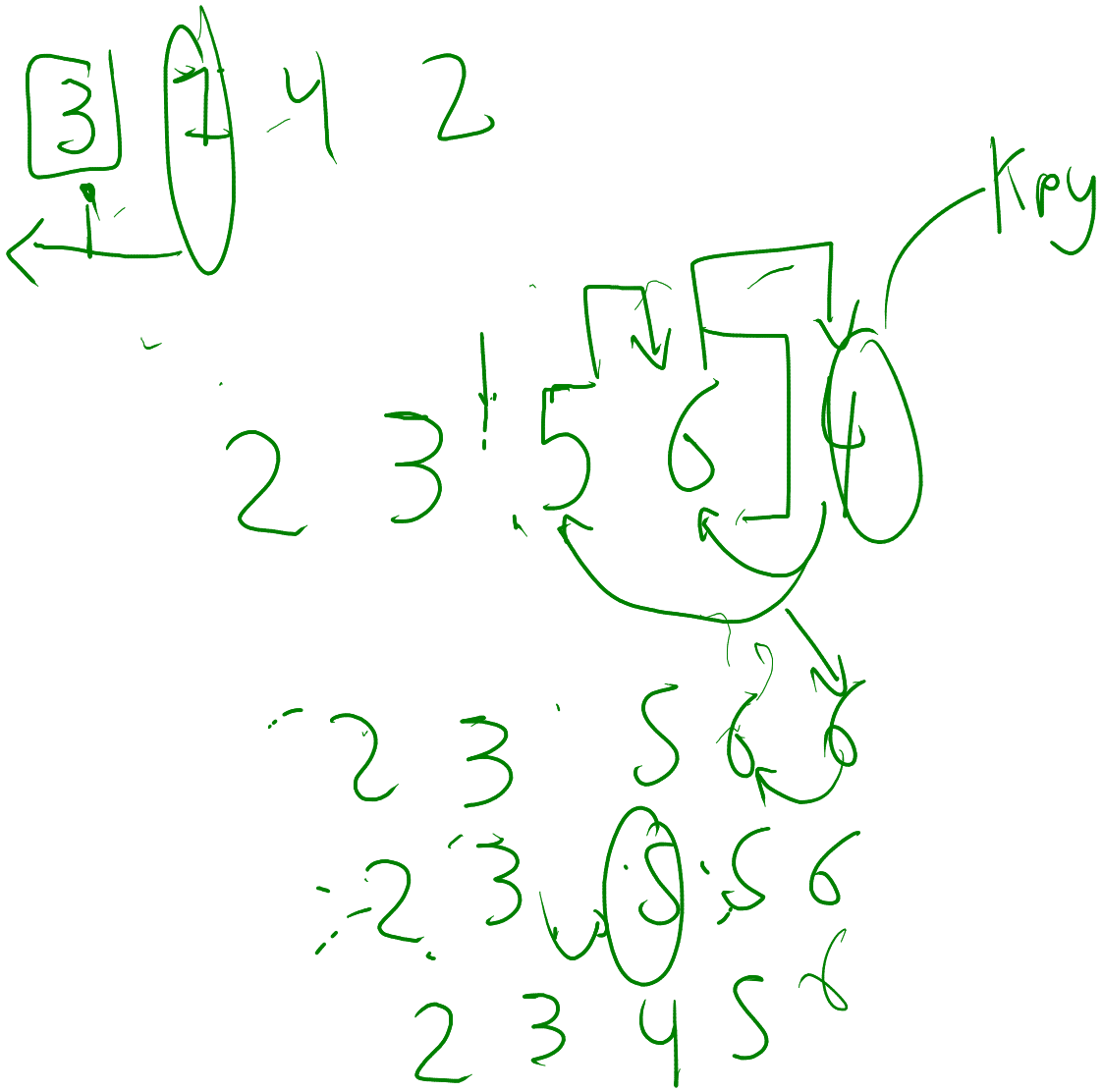

1 4 7 10 3 5 0 3
Valor
1) 3 < 10 ✓
1 4 7 10 10 5 0 3
2) 3 < 7
1 4 7 7 10 5 0 3
3) 3 ≤ 4
1 4 4 7 10 5 0 3

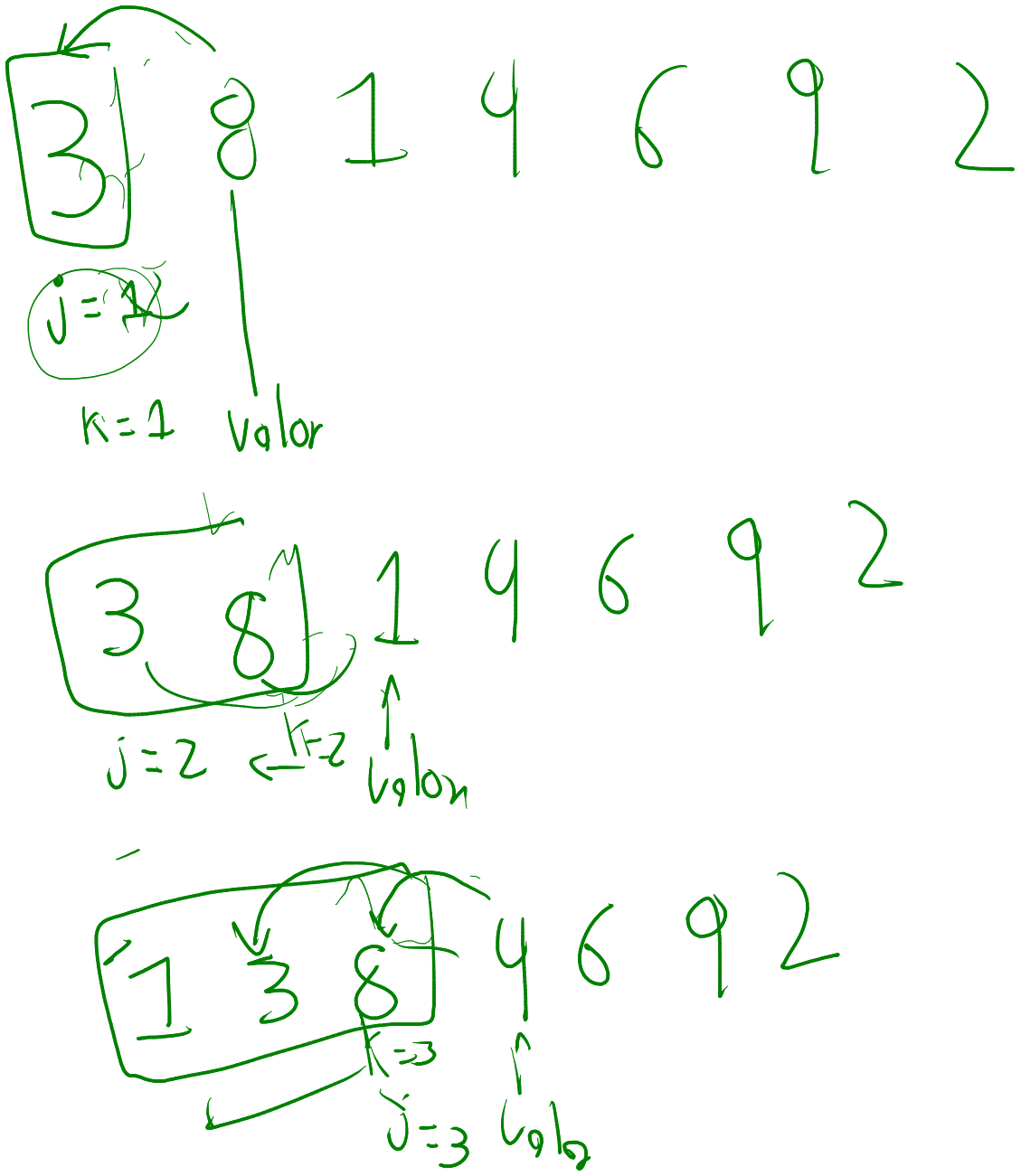

4) 3 < 1
j = 5
Romper
1 3 4 7 10
5 0 3

Ejercicio


5 1 8 7 6 2 3
j = 1
lógica
1 5 8 7 6 2 3
j = 2
1 5 8 7 6 2 3
j = 3
1 5 7 8 6 2 3
j = 4
1 5 6 7 8 2 3
j = 5
1 2 5 6 7 8 3
j = 6
1 2 3 5 6 7 8


7 9 3 1 4 2 6 8 j=1
7 9 3 1 4 2 6 8 j=2
3 7 9 1 4 2 6 8 j=3
1 3 7 9 4 2 6 8 j=4
1 3 4 7 9 2 6 8 j=5
1 2 3 4 7 9 6 8 j=6
1 2 3 4 6 7 9 8 j=7
1 2 3 4 6 7 8 9


3
4
2
key
2
3
5
6
4
2
3
5
6
2
3
5
6
2
3
4
5


3 8 1 4 6 9 2
j = 1
k = 1
valor
3 8 1 4 6 9 2
j = 2
k = 2
valor
1 3 8 4 6 9 2
k = 3
j = 3
valor

Algoritmos de Ordenamiento

Selection sort

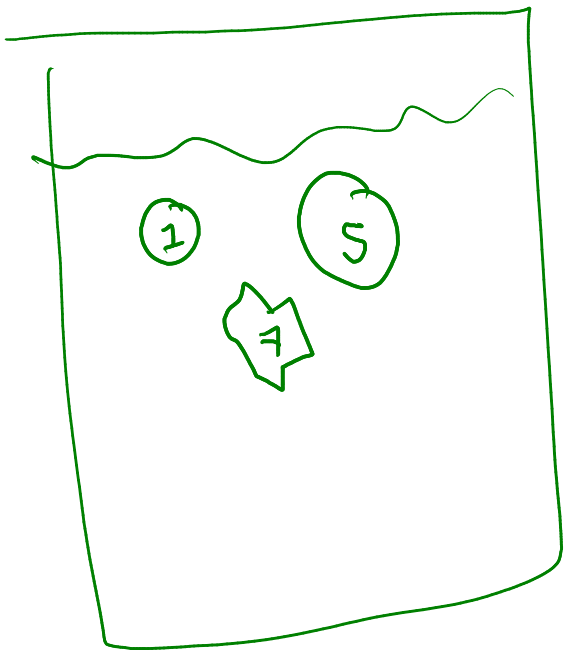
- Considere un arreglo A[1..n]
- Encuentre el menor elemento, colóquelo en la posición 1, ahora considere el arreglo A[2..n]
- Encuentre el menor elemento de A[2..n], colóquelo en la posición 2.
- Repita el proceso hasta llegar al a posición n

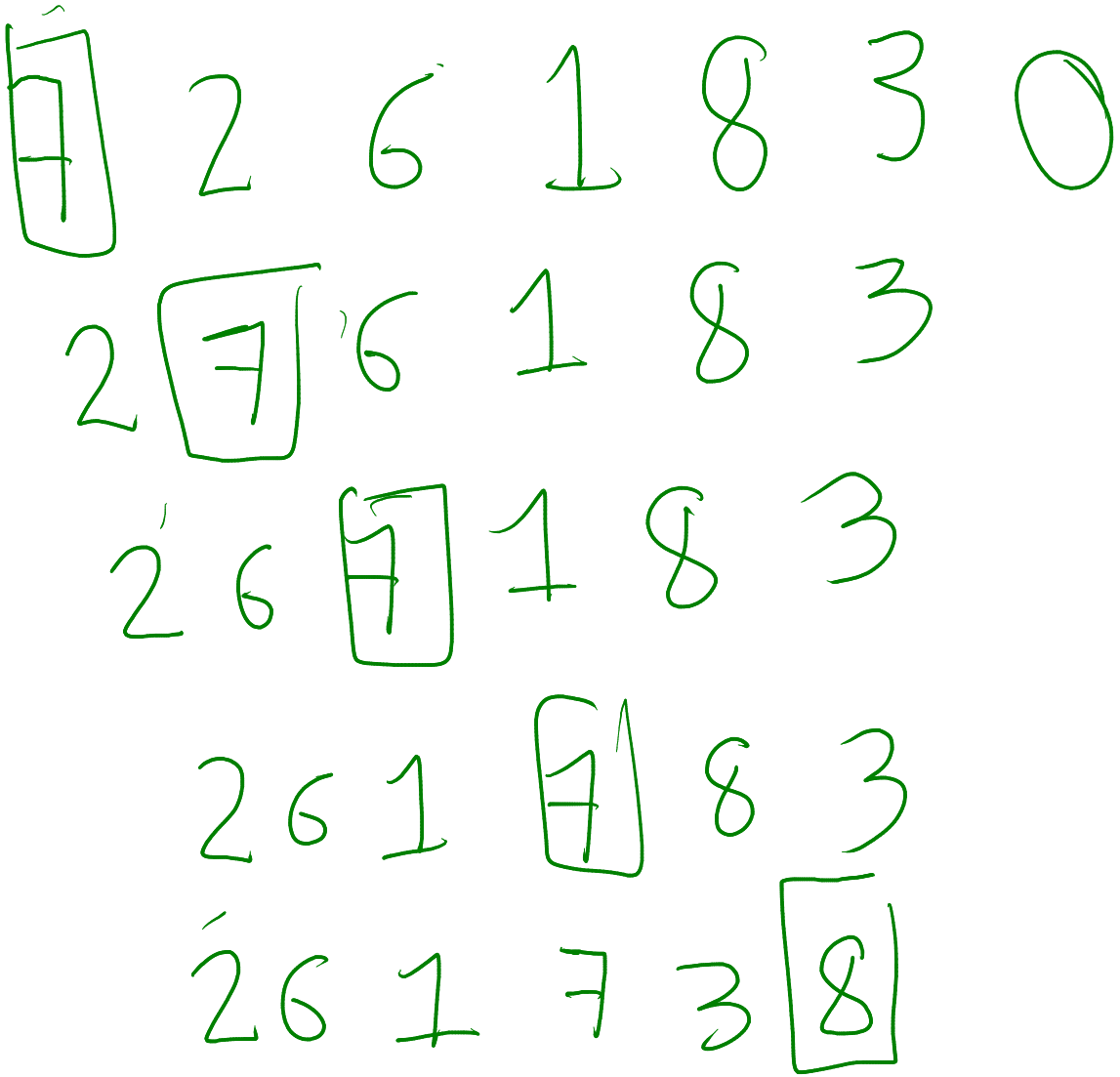
Algoritmos de Ordenamiento

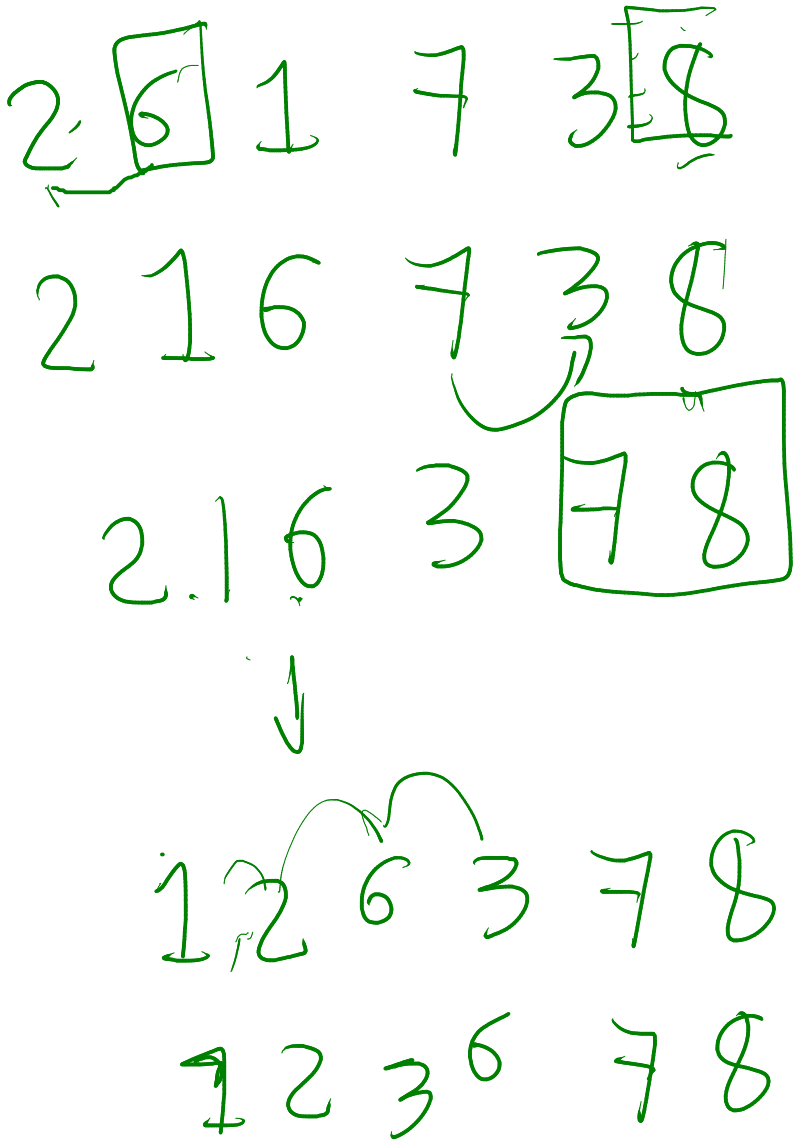
Selection sort

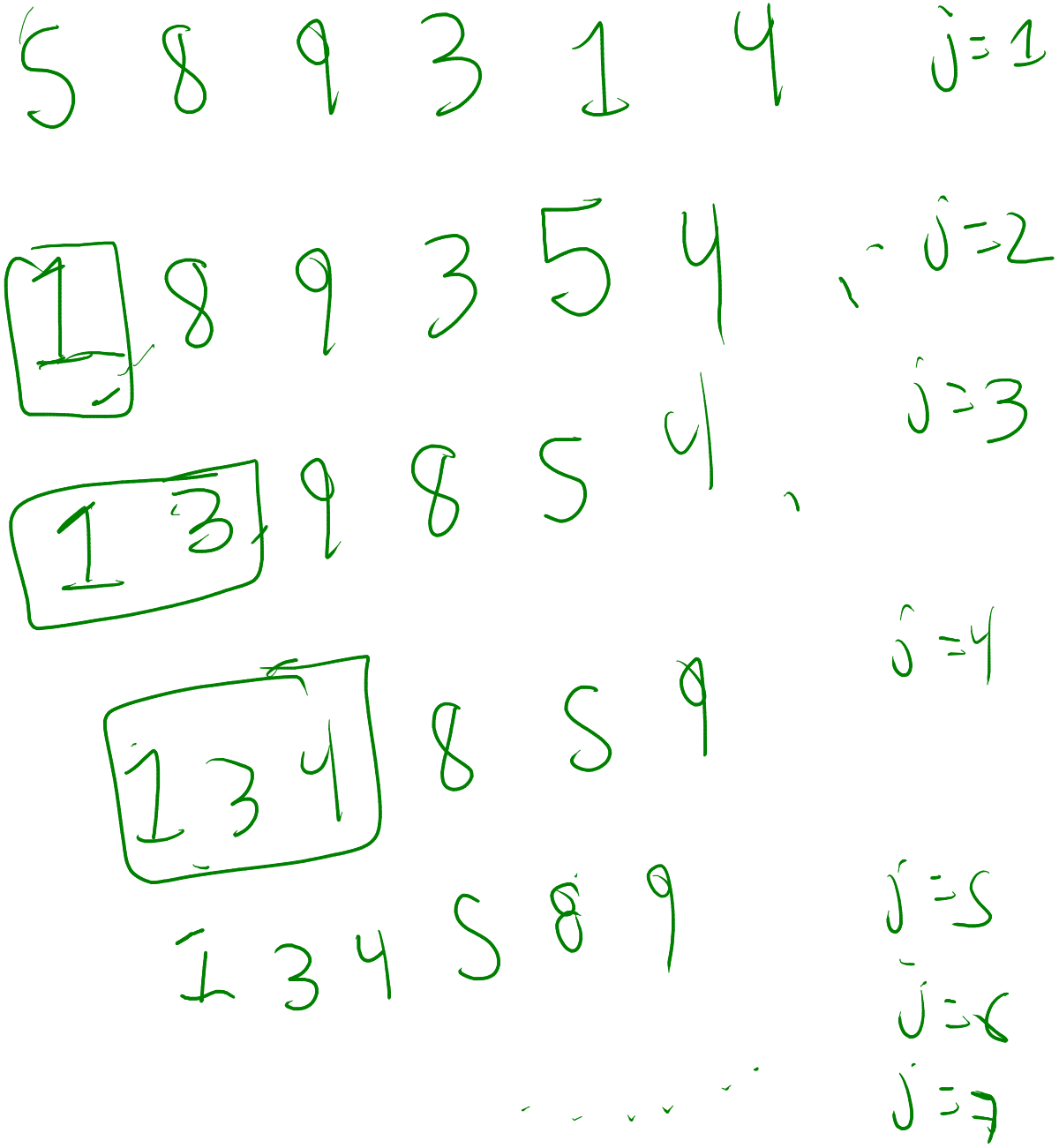
Lista ordenada	Sub lista no ordenada	Menor elemento
()	(11, 25, 12, 22, 64)	11
(11)	(25, 12, 22, 64)	12
(11, 12)	(25, 22, 64)	22
(11, 12, 22)	(25, 64)	25
(11, 12, 22, 25)	(64)	64
(11, 12, 22, 25, 64)	()	

Burbuja








5 8 9 3 1 4 j=1
1 8 9 3 5 4 j=2
1 3 9 8 5 4 j=3
1 3 4 8 5 9 j=4
1 3 4 5 8 9 j=5
j=6
j=7

Referencias

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. Introduction to Algorithms, Third Edition (3rd ed.). The MIT Press. Pages 5-29