Métodos de ordenamiento de datos

Algoritmos y Resolución de problemas

Carreras LCC LSI TUPW - Dto Informática- FCEFyN

Métodos de ordenamiento de datos

Ordenar un arreglo

- Disponer las componentes de un **orden creciente o decreciente** según un determinado criterio numérico, alfabético, alfanumérico, **para facilitar la búsqueda de elementos dentro de él.**

Métodos de ordenamiento de datos

Tipos de ordenamiento: dos tipos internos y externos.

Ordenación INTERNA: La ordenación interna se realiza cuando los **datos están en memoria principal**. Como la memoria principal es de **acceso directo** o aleatorio, la principal **ventaja**, es que se puede **acceder a una determinada componente sin recorrer las anteriores**, por tanto el tiempo de acceso a cualquier elemento del arreglo es siempre el mismo.

Ordenación EXTERNA: La ordenación externa se lleva a cabo cuando el **volumen de los datos** a tratar es demasiado grande y los mismos **no caben en la memoria principal** de la computadora. En estos casos la ordenación **es más lenta** ya que el tiempo que se requiere para acceder a cualquier elemento depende de la última posición a la que se accedió.

Otro criterio de clasificación :

Algoritmos de INTERCAMBIO: En este tipo de algoritmos los elementos se consideran de dos en dos, **se comparan y se INTERCAMBIAN** si no están en el orden requerido. Este proceso se repite hasta que se han analizado todos los elementos.

***Algoritmos de INSERCIÓN*:** Se caracterizan porque los elementos que van a ser ordenados son considerados de a uno a la vez. Cada **elemento es INSERTADO en la posición que le corresponde,** respecto al resto de los elementos ya ordenados de acuerdo al criterio considerado.

***Algoritmos de SELECCIÓN*:** Estos algoritmos se caracterizan porque **se busca o SELECCIONA el elemento más pequeño (o más grande),** según el criterio de ordenamiento elegido, de todo el conjunto de elementos y **se coloca en su posición adecuada**. El proceso se repite para el resto de los elementos hasta que todos son analizados.

los algoritmos más utilizados, clasificados por su tipo

TIPO	NOMBRE
Intercambio	**Burbuja** Quicksort
Selección	**Selección directa.**
Inserción	**Inserción directa**. Inserción binaria Shell Hashing

Método de la Burbuja

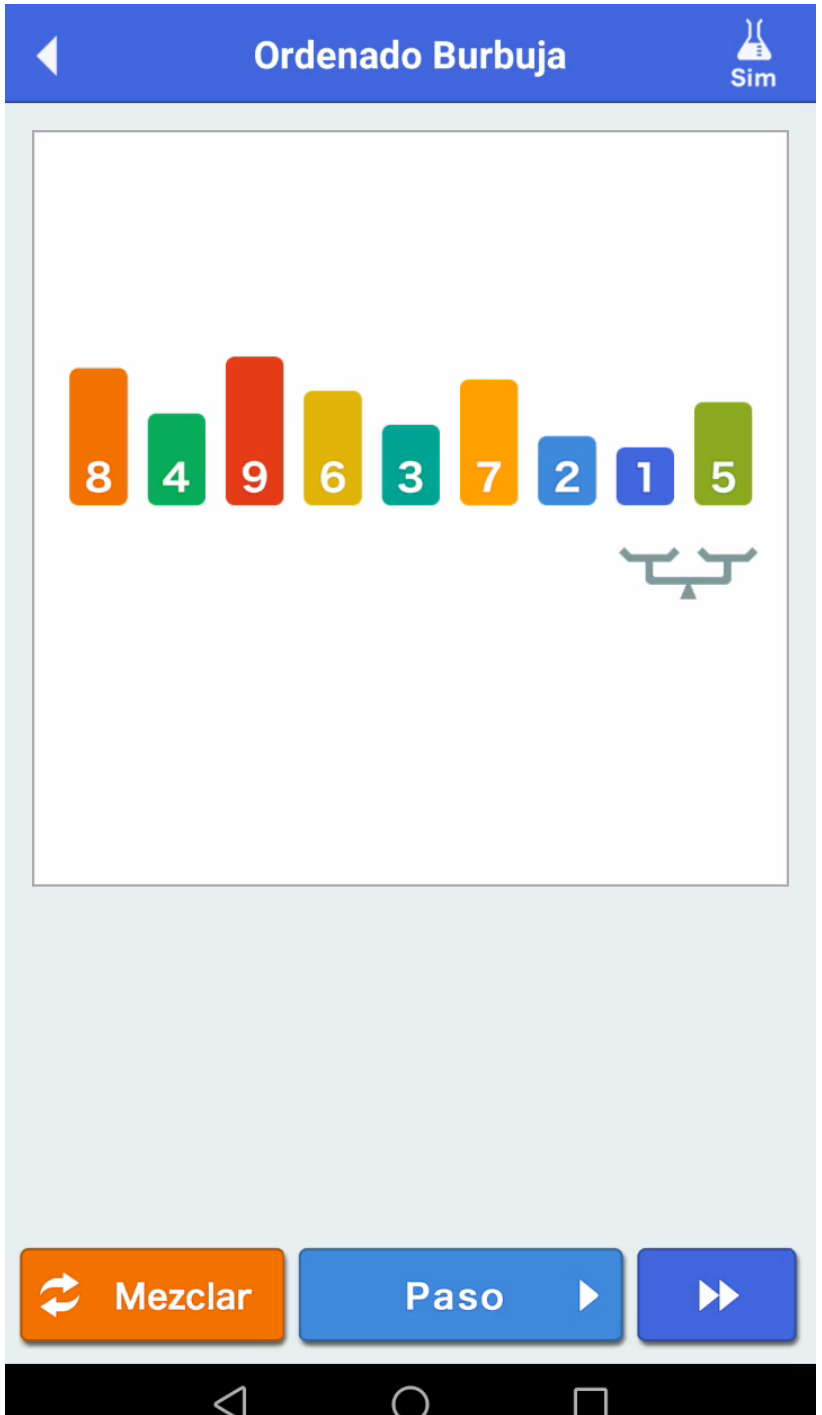
El método se basa en la **comparación de los elementos** adyacentes del arreglo,

intercambiándolos si están desordenados.

De este modo, si el ordenamiento es ascendente,

los valores **más pequeños burbujean hacia las primeras componentes** del arreglo (hacia la primera posición),

mientras que los valores **más grandes se "hunden" hacia el fondo del arreglo**.

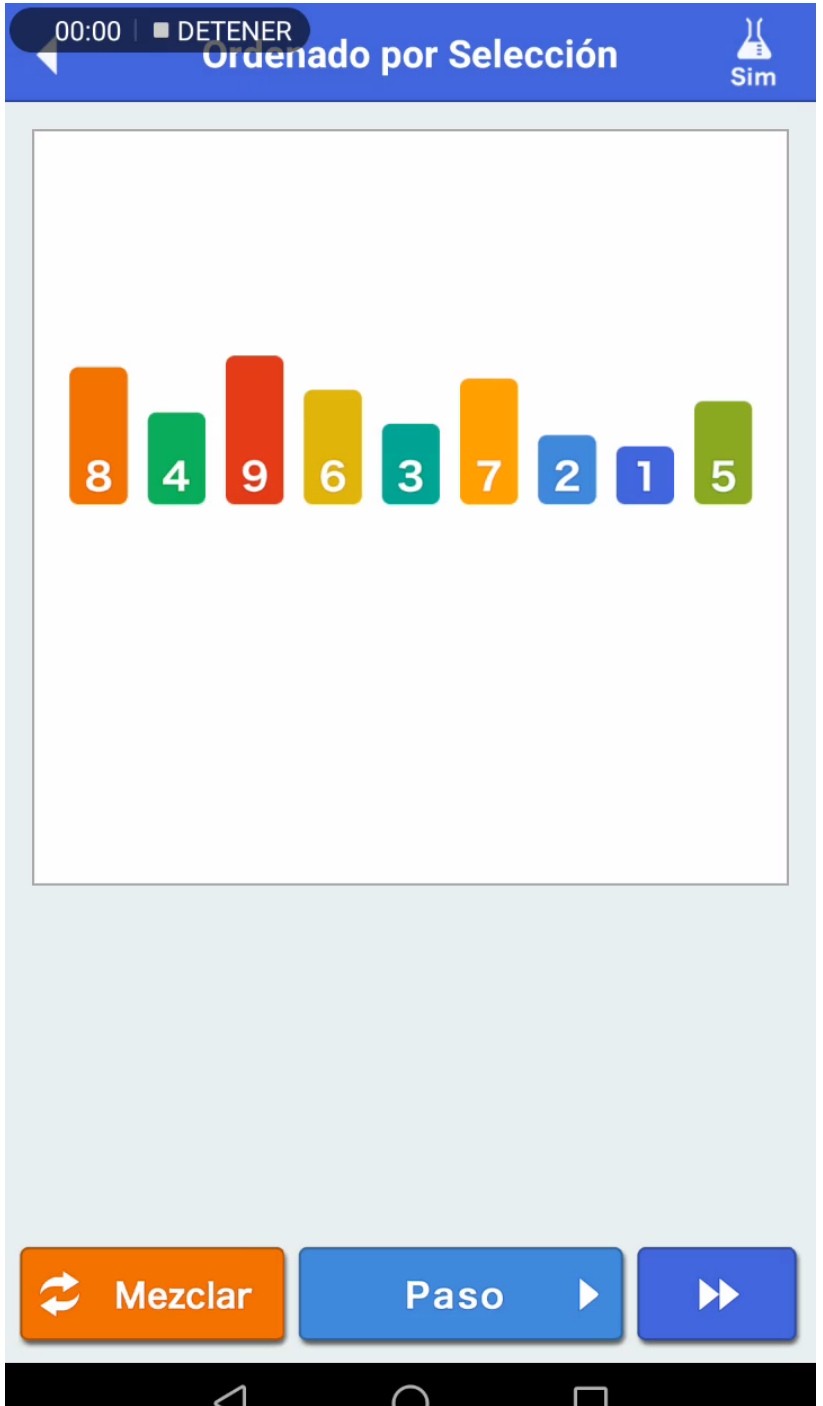


Método de Selección

Este método consiste en **buscar el elemento más pequeño** del arreglo y **ponerlo en la primera posición**; luego,

entre los restantes, **se busca el elemento más pequeño y se coloca en segundo lugar,**

y así sucesivamente hasta colocar el último elemento.

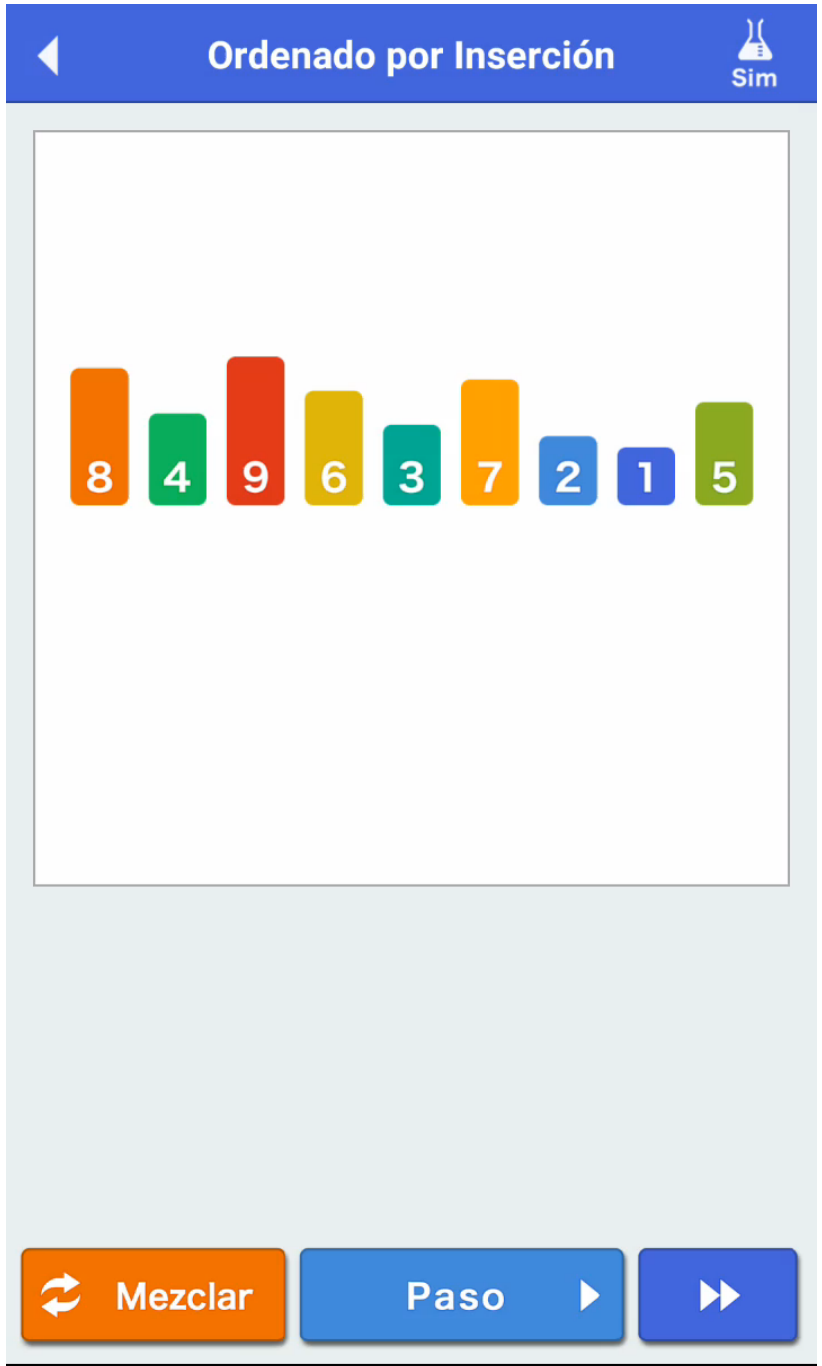


Método de Inserción directa

Este método consiste en **insertar un elemento a[i],**

en el lugar que le corresponde

entre los anteriores a[0]....a[i-1], que ya están ordenados



Método de la Burbuja mejorado

Método de la Burbuja mejorado

40	8	4	9	10	35
a[0]	a[1]	a[2]	a[3]	a[4]	a[5]

Cota = 5

K= 1

K= -1

1 º PASADA por el arreglo desde 0 hasta cota -1 (4)

¿a[i] >a[i+1]) ?

i= 0 **K= 0**

40	8	4	9	10	35
a[0]	a[1]	a[2]	a[3]	a[4]	a[5]

Se intercambian el 8 y el 40. y guarda el valor de i en K

¿a[i] >a[i+1]) ?

i= 1 **K= 1**

8	40	4	9	10	35
a[0]	a[1]	a[2]	a[3]	a[4]	a[5]

Se intercambia el 40 y 4 y guarda el valor de i en K

¿a[i] >a[i+1]) ?

i= 2 **K= 2**

8	4	40	9	10	35
a[0]	a[1]	a[2]	a[3]	a[4]	a[5]

Se intercambia el 9 y el 40.y guarda i en k

¿a[i] >a[i+1]) ?

i= 3 **K= 3**

8	4	9	40	10	35
a[0]	a[1]	a[2]	a[3]	a[4]	a[5]

Se intercambia el 40 y el 10. y se guarda i en K

¿a[i] >a[i+1]) ?

i= 4 **K= 4**

8	4	9	10	40	35
a[0]	a[1]	a[2]	a[3]	a[4]	a[5]

Se intercambia el 35 y el 40. y se guarda i en k

8	4	9	10	35	40
a[0]	a[1]	a[2]	a[3]	a[4]	a[5]

termino la primera pasada

Cota = 4 → **2 º PASADA por el arreglo desde 0 hasta cota -1 (3)**

cota	i	k
5		1
		-1
	0	0
	1	1
	2	2
	3	3
4	4	4

```
void ordenar ( entero a[n] )
Comienzo
entero    k, i , aux, cota
cota= n-1
k= 1
Mientras ( k != -1)
  k = -1

   Para i Desde 0 Hasta cota-1
     Si   (a[i] >a[i+1])
          Entonces  aux = a[i]
                    a[i] = a[i+1]
                    a[i+1] =aux
                     k =i
     FinSi
   FinPara

cota =k

FinMientras

retorna()
Fin
```

/* K guarda la posición donde se realizó el cambio*/

/* Cota guarda el índice del último cambio de la actual pasada como límite superior para la siguiente pasado */

2 º PASADA por el arreglo desde 0 hasta cota -1

Cota = 4

K= -1

¿a[i] >a[i+1]) ?

i= 0

K= 0

8	4	9	10	35	40
a[0]	a[1]	a[2]	a[3]	a[4]	a[5]

Se intercambian. El 8 y el 4 y guarda el valor de i en K

¿a[i] >a[i+1]) ?

i= 1

4	8	9	10	35	40
a[0]	a[1]	a[2]	a[3]	a[4]	a[5]

NO HAY INTERCAMBIO

¿a[i] >a[i+1]) ?

i= 2

4	8	9	10	35	40
a[0]	a[1]	a[2]	a[3]	a[4]	a[5]

NO HAY INTERCAMBIO

¿a[i] >a[i+1]) ?

i= 3

4	8	9	10	35	40
a[0]	a[1]	a[2]	a[3]	a[4]	a[5]

NO HAY INTERCAMBIO

termino la segunda pasada

Cota = 0

Método de la Burbuja mejorado

cota	i	k
4		-1
	0	0
	1	
	2	
0	3	

```
void ordenar ( entero a[n] )
Comienzo
entero    k, i , aux, cota
cota= n-1
k= 1
Mientras ( k != -1)
  k = -1

    Para i Desde 0 Hasta cota-1
      Si   (a[i] >a[i+1])
            Entonces  aux = a[i]
                      a[i] = a[i+1]
                      a[i+1] =aux
                       k =i
                                  /* K guarda la posición donde se realizó el cambio*/
          FinSi
      FinPara

cota =k                           /* Cota guarda el índice del último cambio de la actual pasada como límite superior para la siguiente pasado */

FinMientras

retorna()
Fin
```

3 º PASADA por el arreglo desde 0 hasta cota -1

Método de la Burbuja mejorado

cota	i	k
0		-1

Cota = 0

K= -1

-1

3 º PASADA por el arreglo desde 0 hasta cota -1

4	8	9	10	35	40
a[0]	a[1]	a[2]	a[3]	a[4]	a[5]

No cambia el valor de K

```
void ordenar ( entero a[n] )
Comienzo
entero    k, i , aux, cota
cota= n-1
k= 1
Mientras ( k != -1)
  k = -1
    Para i Desde 0 Hasta cota-1
      Si   (a[i] >a[i+1])
            Entonces  aux = a[i]
                      a[i] = a[i+1]
                      a[i+1] =aux
                       k =i
      FinSi
    FinPara
cota =k
FinMientras

retorna()
Fin
```

Termina el ordenamiento

No se ejecuta ciclo Para

21	4	9	25	10	37	40	45	50	55	60	70
a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]	a[11]

ACTIVIDAD Hacer el seguimiento y responder a las preguntas

cota	i	k

Método de la BURBUJA MEJORADO

```
void ordenar ( entero a[n] )
Comienzo
entero    k, i , aux, cota
cota= n-1
k= 1
Mientras ( k != -1)
  k=-1
  Para i Desde 0 Hasta cota-1

  Si  (a[i] >a[i+1])

                Entonces  aux = a[i]
                          a[i] = a[i+1]
                          a[i+1] =aux
                          k =i

    FinSi
 FinPara
 cota =k
FinMientras
retorna()

Fin
```

Cuántas COMPARACIONES?

Cuántos Movimientos de componentes?

Método de SELECCIÓN

Método de la SELECCION

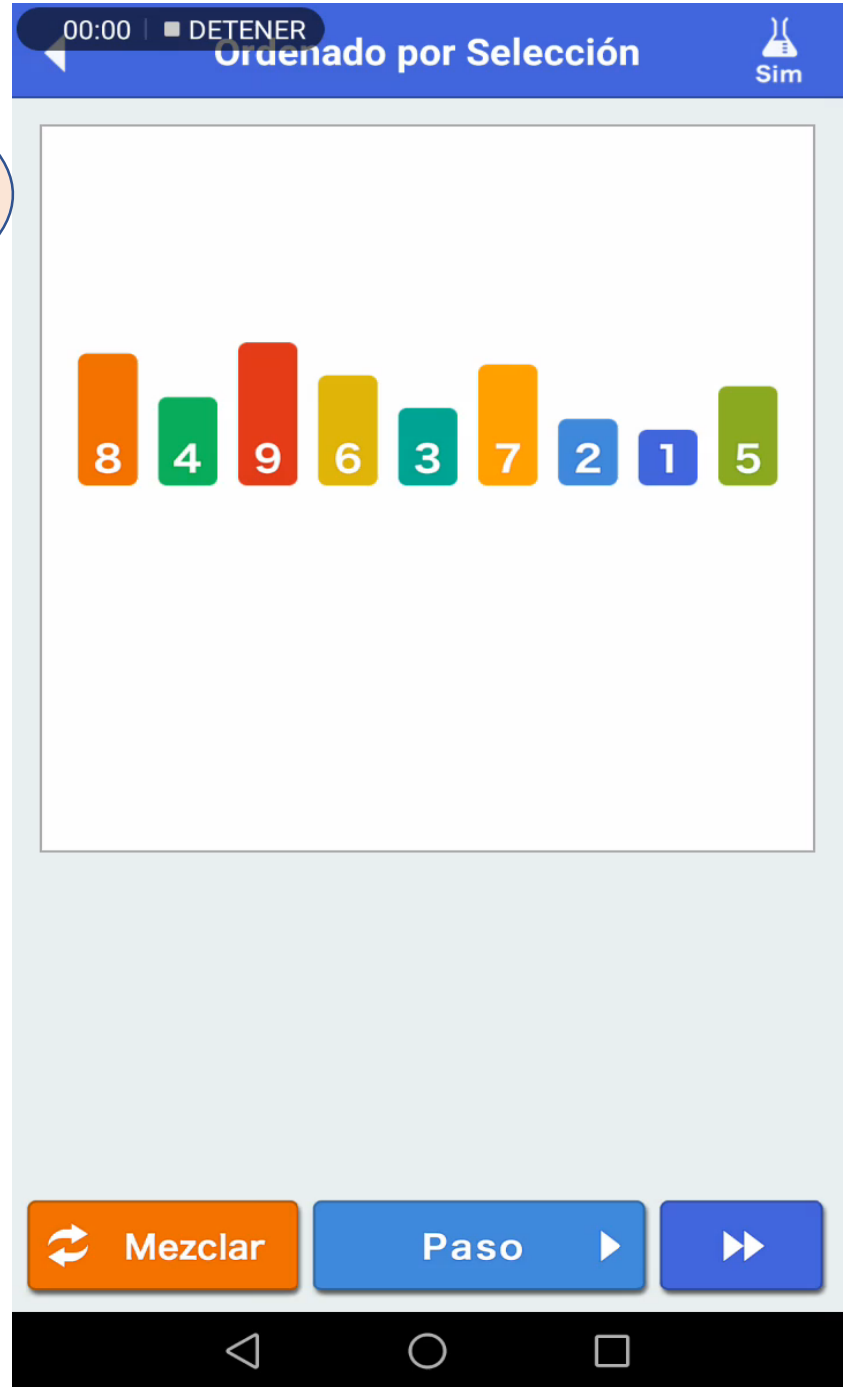
```
void ordenar ( entero a[n] )
Comienzo
entero  i , j, min, aux
  Para i Desde 0 Hasta  n-2
      min=i
      Para j Desde  i+1 hasta  n -1
          Si (a [ j] < a [min])
              Entonces     min =j
           FinSi
      FinPara
      aux= a [i]
      a [i] = a [min]
      a [ min] = aux
FinPara
retorna()
Fin
```

Cuántas COMPARACIONES?

/*busca el índice donde se ubica el mínimo entre los elementos a[i] .....a[n-1] */

/*intercambia los valores */

Cuántas Movimientos de componentes?



ACTIVIDAD Hacer el seguimiento y responder a las preguntas

Método de SELECCIÓN

21	4	9	25	10	37	40	45	50	55	60	70
a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]	a[11]

i	j	min

```
void ordenar ( entero a[n] )
Comienzo
entero  i , j, min, aux
  Para i Desde 0 Hasta  n-2
      min=i
      Para j Desde  i+1 hasta  n -1
          Si (a [ j] < a [min])
              Entonces     min =j
           FinSi
      FinPara
        aux= a [i]              /*intercambia los valores */
        a [i] = a [min]
        a [ min] = aux
FinPara
retorna()
Fin
```

Método de INSERCIÓN DIRECTA

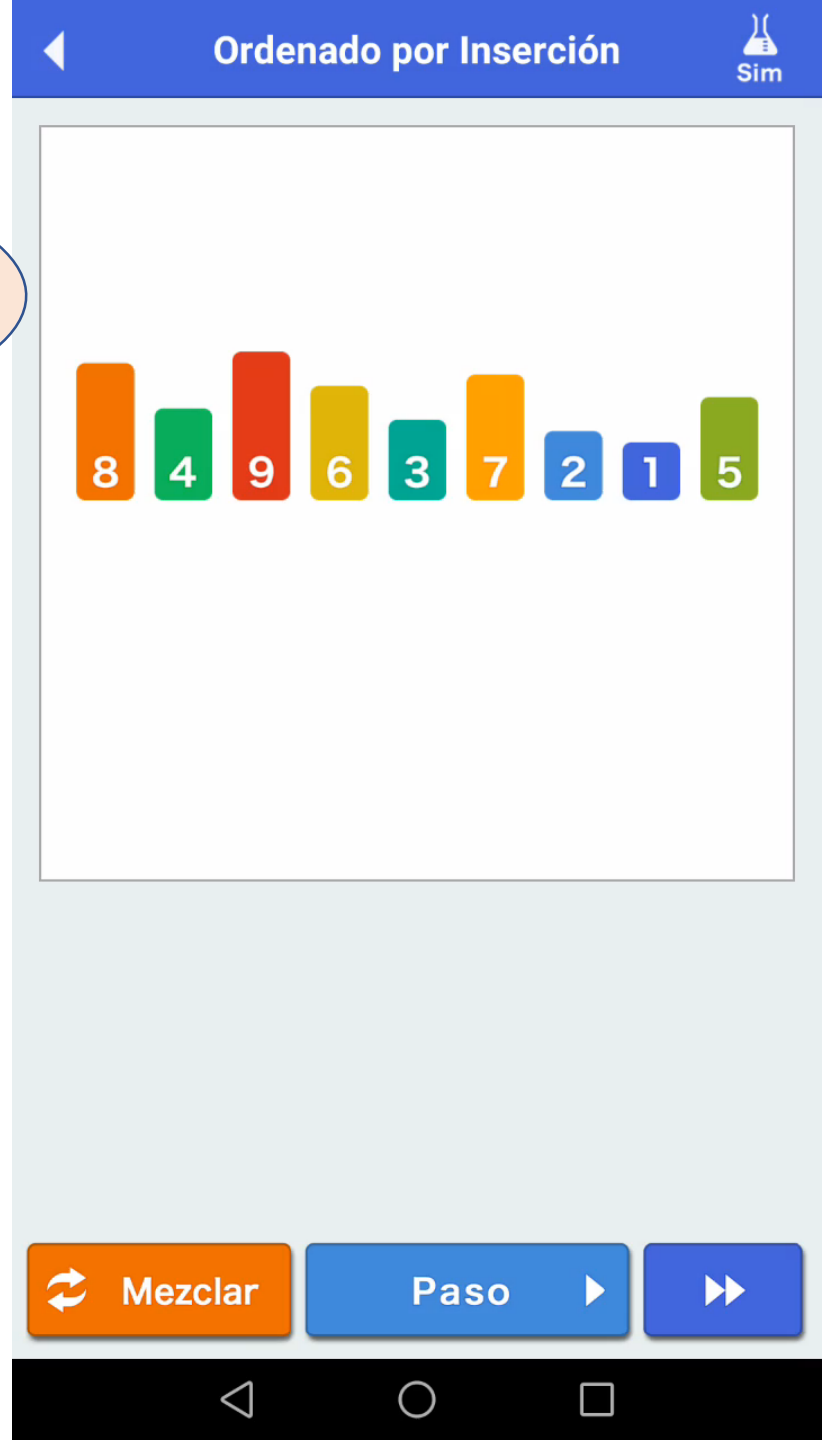
Método de INSERCIÓN DIRECTA

```
void ordenar ( entero a[n] )
Comienzo
entero  i , j ,valor
Para i Desde 1 Hasta  n-1
      valor  = a[i]
       j= i-1

      Mientras (( j ≥ 0 ) y ( valor < a [ j ] ))
         a [ j+1]= a [j]      /* los valores se corren un lugar a la derecha */

            j=j-1
      FinMientras

      a[j+1] = valor

FinPara
retorna()
Fin
```

Cuántas COMPARACIONES?

Cuántos Movimientos de componentes?

21	4	9	25	10	37	40	45	50	55	60	70
a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]	a[11]



ACTIVIDAD Hacer el seguimiento y responder a las preguntas

Método de INSERCIÓN

i	j	VALOR

21	4	9	25	10	37	40	45	50	55	60	70
a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]	a[11]

```
Comienzo
entero  i , j ,valor
Para i Desde 1 Hasta  n-1
      valor  = a[i]
       j= i-1
  Mientras (( j ≥ 0 ) y ( valor < a [ j ] ))
       a [ j+1]= a [j]      /* los valores se corren un lugar a la derecha */
       j=j-1
    FinMientras
    a[j+1] = valor
FinPara
retorna()
Fin
```