Taller de Programación

AGENDA

Método de ordenación: inserción

ARREGLOS – Ordenación - INSERCION

Este algortimo consta de N vueltas, donde N es la dimension lógica del arreglo. La idea general de este algoritmo es ir considerando subconjuntos de datos del vector e ir ordenándolos.

En la **primera Vuelta,** se trabaja el subconjunto formado por el primer elemento del arreglo que obviamente se considera ordenado.

En la **segunda vuelta**, se trabaja el subconjunto formado por el primer y segundo elemento del arreglo y se ordena ese subconjunto de manera de encontrar en que posición debe estar el segundo elemento para que el arreglo siga ordenado.

En la **tercera vuelta**, se trabaja el subconjunto formado por el primer, segundo y tercer elemento del arreglo y se ordena ese subconjunto de manera de encontrar en que posición debe estar el tercer elemento para que el arreglo siga ordenado (ya se sabe que el primero y segundo están ordenados).

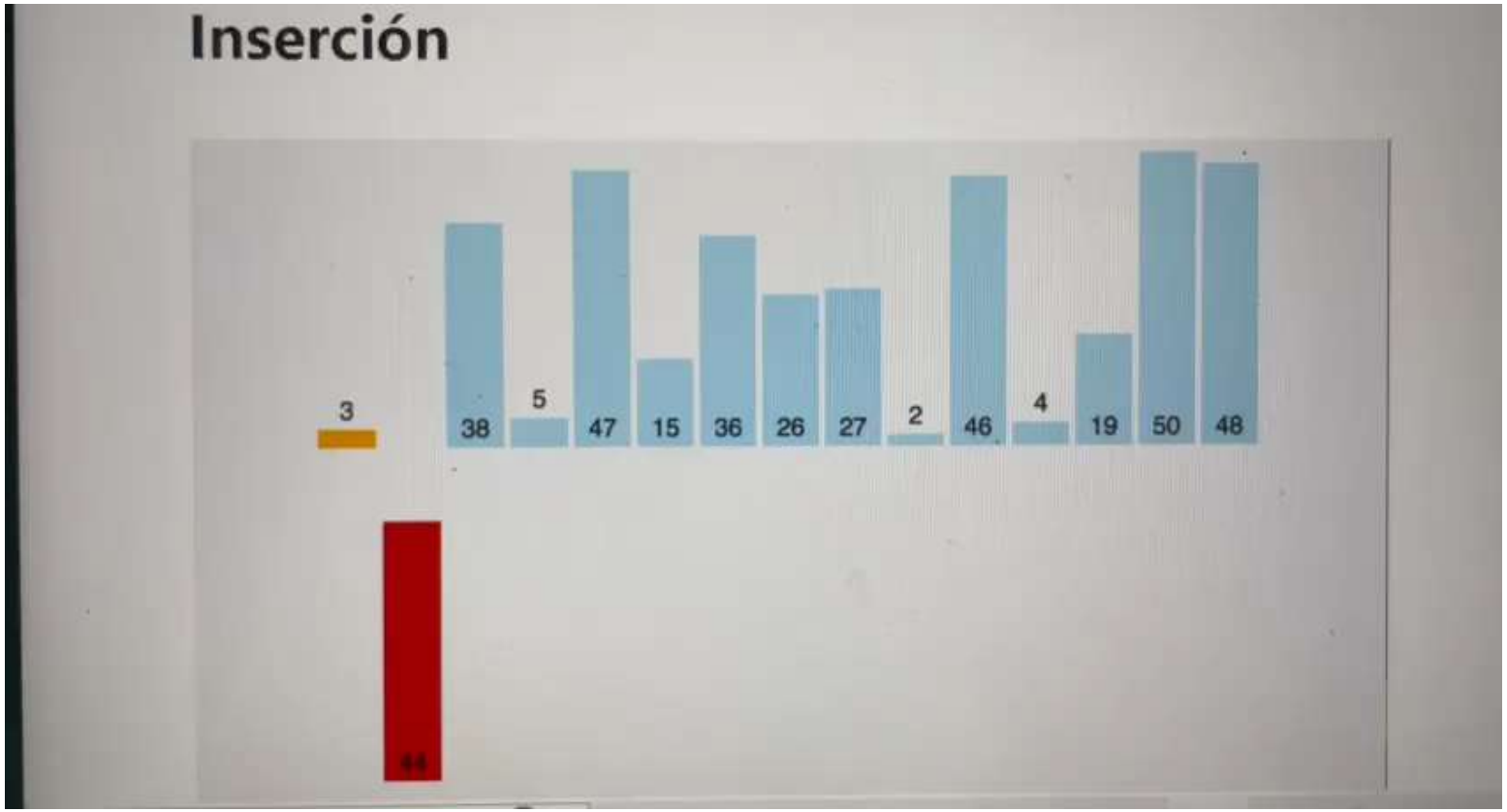
En la **cuarta vuelta**, se trabaja el subconjunto formado por el primer, segundo, tercer y cuarto elemento del arreglo y se ordena ese subconjunto de manera de encontrar en que posición debe estar el cuarto elemento para que el arreglo siga ordenado (ya se sabe que el primero, el segundo y el tercero están ordenados).

Esto se repite hasta recorrer todo el arreglo, dando así un total de N vueltas.

ARREGLOS – Ordenación - INSERCION

QUE NECESITAMOS CONOCER?

- Dimensión lógica del arreglo.
- Posición que se debe comparar
- Cuántos elementos ya se encuientran ordenados



ARREGLOS – Ordenación - INSERCION

5	3	2	1	4	6	

VUELTA 1

Tomo ele elemento ubicado en la posición 2 (3) y se compara desde la posición 1 hasta la 1 para ver en qué posición debe insertarse.

5	3	2	1	4	6	

Debe insertarse en la posición 1

Se encuentra que el valor debe insertarse en la posición 1, por lo tanto, se realiza un corrimiento desde la posición 1 hasta la 2 para "hacer" lugar en el vector.

5	3	2	1	4	6	

3

ARREGLOS – Ordenación - INSERCION

3	5	2	1	4	6	

dimF = 7
dimL= 6

VUELTA 2

Tomo ele elemento ubicado en la posición 3 (2) y se compara desde la posición 1 hasta la 2 para ver en qué posición debe insertarse.

3	5	2	1	4	6	

Debe insertarse en la posición 1

Se encuentra que el valor debe insertarse en la posición 1, por lo tanto, se realiza un corrimiento desde la posición 1 hasta la 3 para "hacer" lugar en el vector.

3	5	2	1	4	6	

2

ARREGLOS – Ordenación - INSERCION

dimF = 7
dimL= 6

VUELTA 3

Tomo ele elemento ubicado en la posición 4 (1) y se compara desde la posición 1 hasta la 3 para ver en qué posición debe insertarse.

Debe insertarse en la posición 1

Se encuentra que el valor debe insertarse en la posición 1, por lo tanto, se realiza un corrimiento desde la posición 1 hasta la 3 para "hacer" lugar en el vector.

1

ARREGLOS – Ordenación - INSERCION

dimF = 7
dimL= 6

VUELTA 4 Tomo ele elemento ubicado en la posición 5 (4) y se compara desde la posición 1 hasta la 4 para ver en qué posición debe insertarse.

Debe insertarse en la posición 4

Se encuentra que el valor debe insertarse en la posición 4, por lo tanto, se realiza un corrimiento desde la posición 4 hasta la 5 para "hacer" lugar en el vector.

4

ARREGLOS – Ordenación - INSERCION

dimF = 7
dimL= 6

VUELTA 5 Tomo ele elemento ubicado en la posición 6 (6) y se compara desde la posición 1 hasta la 5 para ver en qué posición debe insertarse.

Debe insertarse en la posición 6

Se encuentra que el valor debe insertarse en la posición 6, por lo tanto, se realiza un corrimiento desde la posición 6 hasta la 6 para "hacer" lugar en el vector.

6

ARREGLOS – Ordenación - INSERCION

```
Program ordenar;

Const  dimF =... {máxima longitud del arreglo}

Type
      TipoElem = ... { tipo de datos del vector }
      Indice   =  0.. dimF;
      Tvector =  array  [ 1..dimF] of TipoElem;

Var
  a:Tvector;
  dimL:integer;

Begin
  cargarVector (a, dimL);
  insercion (a, dimL);
End.
```

ARREGLOS – Ordenación - INSERCION

```
Procedure insercion ( var v: tVector; dimLog: indice );
Var
 i, j: indice; actual: tipoElem;

begin
 for i:= 2 to dimLog do begin
     actual:= v[i];
     j:= i-1;
     while (j > 0) y (v[j] > actual) do
       begin
         v[j+1]:= v[j];
         j:= j - 1;
       end;
     v[j+1]:= actual;
 end;
end;
```

ARREGLOS – Ordenación - INSERCION

```
Program ordenar;

Const  dimF = 200

Type
      Indice  =  0.. dimF;
      vectorEnteros =  array  [ 1..dimF] of integer;

Var
  a:vectorEnteros;
  dimL:integer;

Begin
  cargarVector (a, dimL);
  insercion (a, dimL);
End.
```

ARREGLOS – Ordenación - INSERCION

```
Procedure insercion ( var v: vectorEnteros; dimLog: indice );
Var
 i, j: indice; actual: integer;

begin
 for i:= 2 to dimLog do begin
     actual:= v[i];
     j:= i-1;
     while (j > 0) y (v[j] > actual) do
       begin
         v[j+1]:= v[j];
         j:= j - 1;
       end;
     v[j+1]:= actual;
 end;
end;
```

ARREGLOS – Ordenación - INSERCION

CONSIDERACIONES

- Tiempo de ejecución.
- Facilidad para la escritura del mismo.
- Memoria utilizada en su ejecución.
- Complejidad de las estructuras auxiliares que necesite.
- Requiere el mismo tiempo si los datos ya están ordenados, si están al azar, si se encuentran en el orden exactamente inverso al que yo los quiero tener.
- N2
- No estan fácil de implementar.
- El arreglo y variables.
- No require.
- Si los datos están ordenados de menor a mayor el algoritmo solo hace comparaciones, por lo tanto, es de orden (N). Si los datos están ordenados de mayor a menor el algoritmo hace todas las comparaciones y todos los intercambios, por lo tanto, es de orden (N^2). comparaciones.