Tema 5
Métodos de ordenamiento

METODOS DE ORDENAMIENTO INTERNO

Subtemas:

5.1 Algoritmos de ordenamiento internos

5.1.1 Burbuja

5.1.2 Quicksort

5.1.3 ShellSort

5.1.4 Radix

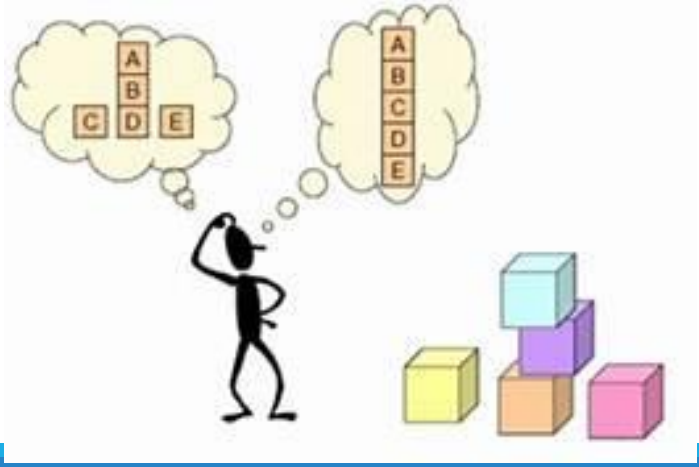
5.2 Algoritmos de ordenamiento externos

5.2.1 Intercalación

5.2.2 Mezcla Directa

5.2.3 Mezcla Natural

¿Qué significa ordenar?



ORDENAR significa reagrupar un conjunto de datos u objetos en una secuencia específica.

- ¿para qué?

CLASIFICACION DE LOS METODOS DE ORDENAMIENTO

Métodos de ordenamiento interno

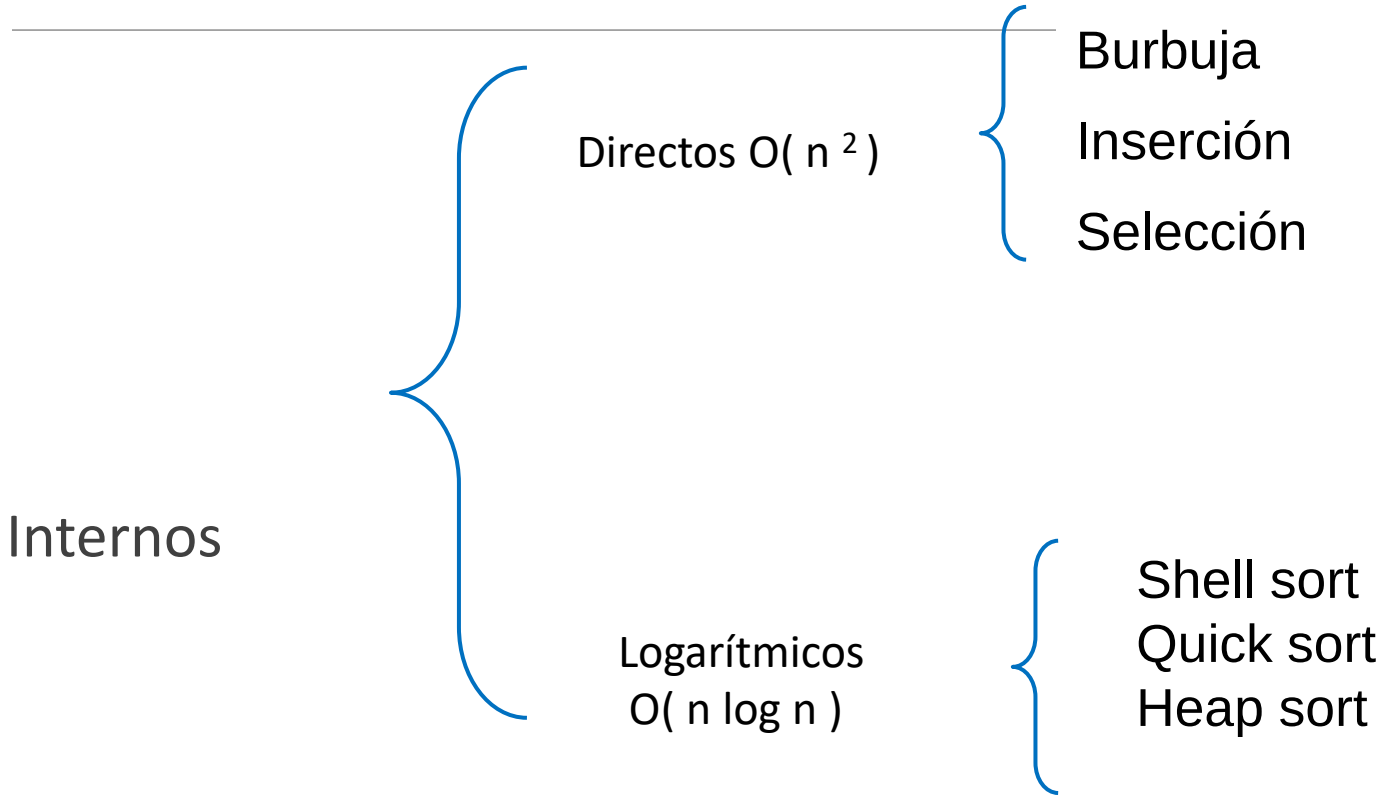
Actúan sobre conjuntos de datos que residen completamente en memoria principal

Métodos de ordenamiento externo

Trabajan con conjuntos de datos que no residen de manera completa en memoria principal.

Internos ---→ arreglos

Externos ----→ archivos


Internos
Directos O(n 2)
Burbuja
Inserción
Selección
Logarítmicos
O(n log n)
Shell sort
Quick sort
Heap sort

Externos

- Intercalación
- Mezcla directa
- Mezcla equilibrada

Ordenamiento por Burbuja simple

El método de la burbuja es uno de los mas simples. Consiste en comparar pares de elementos contiguos de una lista, si se cumple que uno es mayor o menor a otro, entonces los intercambia de posición.

Burbuja simple- código en Java

```
static void burbuja (int a[])
{
int i, j;
/* se realizan n−1 pasadas */
for (i = 0 ; i <= a.length-1 ; i++)
 for (j = 0 ; j < a.length-1 ; j++)
  if (a[ j ] > a[ j+1])
  {
     int aux;
     aux = a[ j];
     a[ j ] = a[ j+1 ];
     a[ j+1 ]= aux ;
   }
}
```

50	26	7	9	15	27

Primera pasada:

26	**50**	7	9	15	27
26	7	**50**	9	15	27
26	7	9	**50**	15	27
26	7	9	15	**50**	27
26	7	9	15	27	**50**

a[j] > a[j+1], intercambia a[j] y a[j+1]
50 > 26, se intercambian 50 y 26

50 > 7, se intercambian 50 y 7

50 > 9, se intercambian 50 y 9

50 > 15, se intercambian 50 y 15

50 > 27, se intercambian 50 y 27

El elemento mayor fue llevado a la ultima posición

Burbuja simple - Ejemplo gráfico

50	26	7	9	15	27

a[j] > a[j+1], intercambia a[j] y a[j+1]
50 > 26, se intercambian 50 y 26

Primera pasada:

26	**50**	7	9	15	27
26	7	**50**	9	15	27
26	7	9	**50**	15	27
26	7	9	15	**50**	27
26	7	9	15	27	**50**

50 > 7, se intercambian 50 y 7

50 > 9, se intercambian 50 y 9

50 > 15, se intercambian 50 y 15

50 > 27, se intercambian 50 y 27

El elemento mayor fue llevado a la ultima posición

Segunda pasada

7	**26**	9	15	27	50
7	9	**26**	15	27	50
7	9	15	**26**	27	50

26 > 7, se intercambian 26 y 7

26 > 9, se intercambian 26 y 9

26 > 15, se intercambian 26 y 15

El 2º y 3er elementos mayores son llevados a su posición definitiva

En este ejemplo el arreglo queda ordenado en la segunda pasada.
pero los dos bucles for seguirán ejecutándose hasta el final.

Ordenación por el método de intercambio directo con señal

Este método es una modificación del método de intercambio directo analizado en la sección anterior. La idea central de este algoritmo consiste en utilizar una marca o señal para indicar que no se ha producido ningún intercambio en una pasada. Es decir, se comprueba si el arreglo está totalmente ordenado después de cada pasada, terminando su ejecución en caso afirmativo. El algoritmo de ordenación por el método de la burbuja con señal es:

Algoritmo 8.3 Burbuja_señal

Burbuja_señal (*A*, *N*)

{El algoritmo ordena los elementos del arreglo utilizando el método de la burbuja con señal. *A* es un arreglo unidimensional de *N* elementos}
{*I*, *J* y AUX son variables de tipo entero. BAND es una variable de tipo booleano}

1. Hacer $I \leftarrow 1$ y BAND ← FALSO
2. *Mientras* (($I \leq N - 1$) y (BAND = FALSO)) *Repetir*
   Hacer BAND ← VERDADERO
   2.1 *Repetir* con *J* desde 1 hasta $N - 1$
      2.1.1 *Si* ($A[J] > A[J + 1]$) *entonces*
         Hacer AUX ← $A[J]$, $A[J] \leftarrow A[J + 1]$, $A[J + 1] \leftarrow$ AUX
         y BAND ← FALSO
      2.1.2 {Fin del condicional del paso 2.1.1}
   2.2 {Fin del ciclo del paso 2.1}
   Hacer $I \leftarrow I + 1$
3. {Fin del ciclo del paso 2}

Algoritmo burbuja meiorado

```
algoritmo Burbuja2

inicio
// Ordenaciones
i ← 1   // Iniciar ordenaciones
ord ← 0 // Iniciar indicador de vector ordenado
mientras ord = 0 hacer
        ord ← 1
        // Comparaciones
        desde j ← 1 hasta n-i hacer
                si elemento[j] > elemento[j+1] entonces
                        // Intercambiar los elementos
                        aux ← V[j]
                        V[j] ← V[j+1]
                        V[j+1] ← aux
                        ord ← 0
                fin_si
        fin_desde
        i ← i+1
fin_mientras
fin
```

análisis

El tiempo de ejecución promedio del algoritmo de ordenamiento por burbuja es del orden $O(n^2)$.

- *Es uno de los algoritmos menos eficientes de ordenación en cuanto a tiempo de ejecución, solamente es recomendable su uso para ordenar un número pequeño de elementos.*

Ordenamiento por Shellsort
método de intervalos decrecientes

El método se denomina así en honor de su inventor Donald Shell.

Este método mejora el ordenamiento por inserción comparando elementos separados por un espacio de varias posiciones. Esto permite que un elemento haga “saltos más grandes" hacia su posición esperada. Los pasos múltiples sobre los datos se hacen con tamaños de espacio cada vez más pequeños.

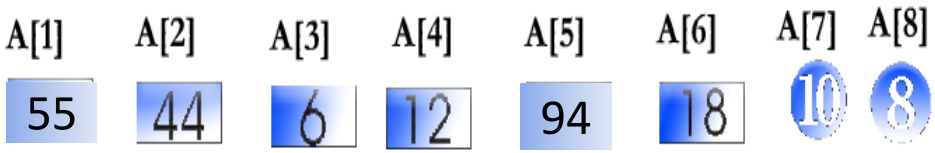
El último paso del Shellsort es un simple ordenamiento por inserción, pero para entonces, los datos están casi ordenados.

Algoritmo Shell

```
Intervalo = n div 2
Mientras (intervalo  > 0 ) hacer
  Desde i = (intervalo + 1) hasta n hacer
    J = i - intervalo
   Mientras (j >0 ) hacer
      K = j + intervalo
      Si  ( a [j] <= a[k])
      Entonces
       J = 0
      Si no
        Intercambio (a[j], a[k]
      Fin si
      j = j - intervalo
   Fin mientras
  Fin desde
  Intervalo = intervalo div 2
Fin mientras
```

Método Shell

Lista original sin ordenar:



1.Se divide la lista original (8 elementos, en este ejemplo) intervalo: 8/2 =4.

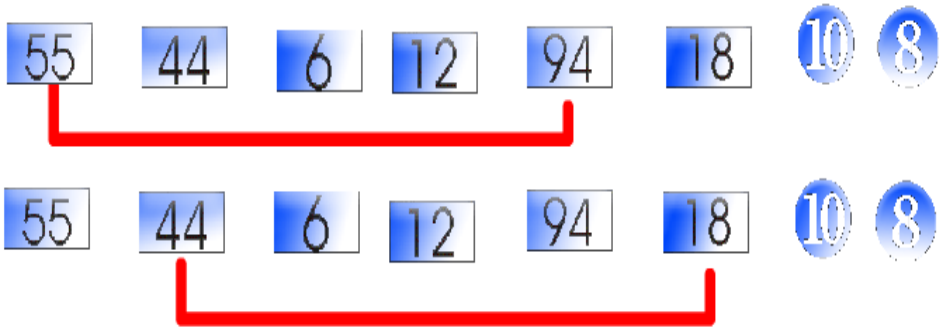
2.Se clasifica cada grupo por separado (se comparan las parejas de elementos y si no estan ordenados se intercambian entre sí de posiciones).

3.Se divide ahora la lista (intervalo o salto de 4/2 = 2) y nuevamente se clasifica cada grupo por separado.

4.Un tercer paso divide nuevamente la lista (intervalo 2/2 =1)clasifica los grupos y completa el trabajo clasificando todos los 8 registros.

Ejemplo

Se inician las comparaciones con tamaño de intervalo 4

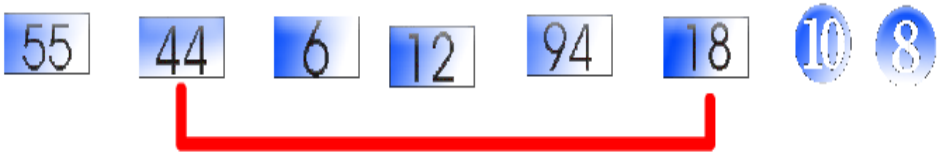


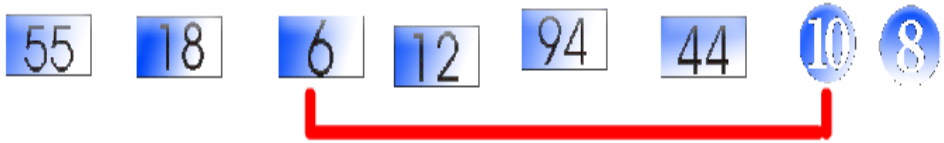
A[1]>A[5], NO
No hay intercambio

A[2]>A[6], SI,
hay intercambio
aux= A[2] ; 44
A[2]= A[6] ; 18
A[6]=aux;

comparaciones con tamaño de intervalo 4

A[2]>A[6], Si
aux=A[2];
A[2]=A[6];
A[6]=aux;



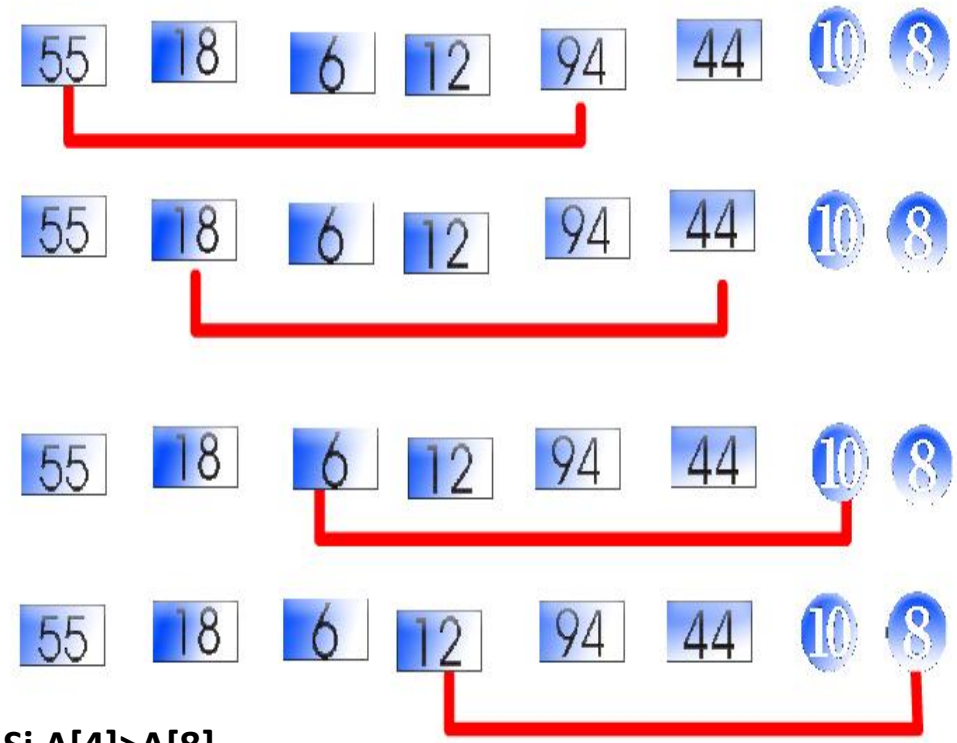


No A[3]>A[7]

No hay intercambio

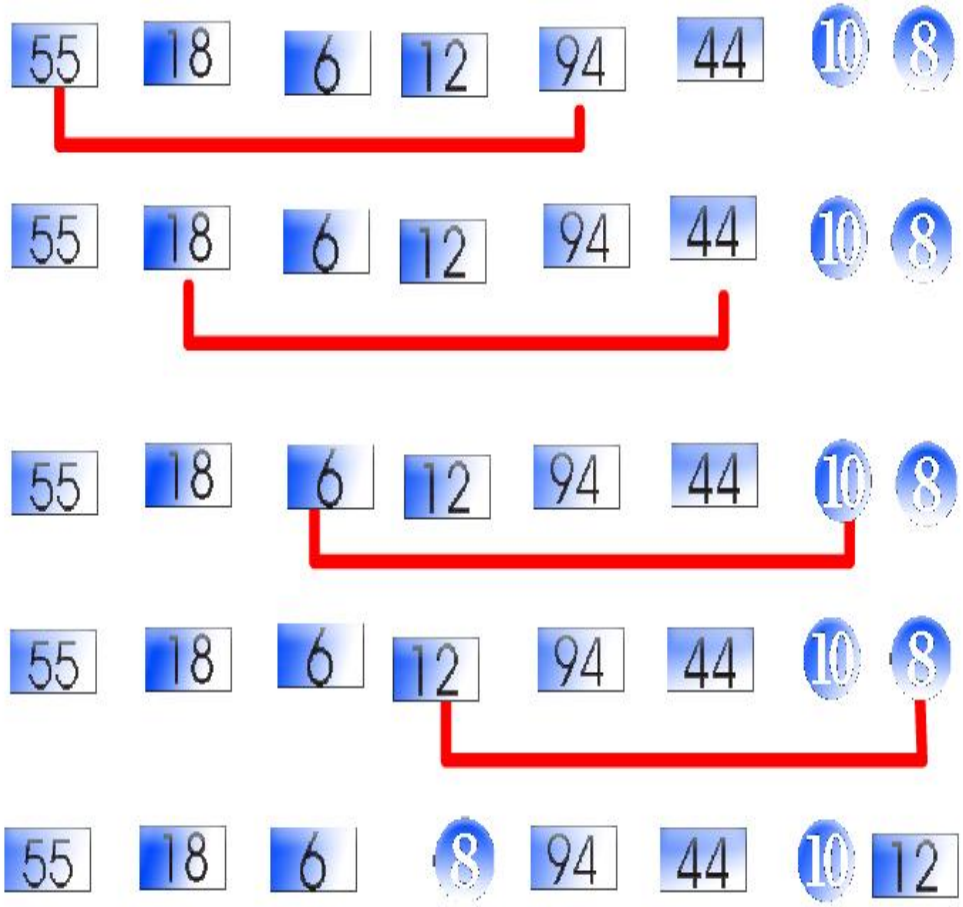
A[3]=6

A[7]=10



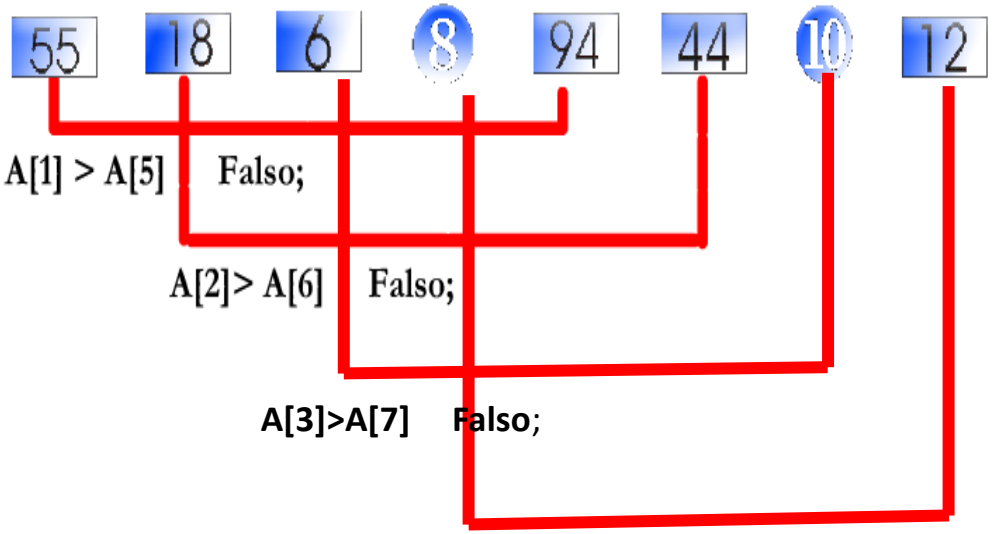
Si A[4]>A[8]
aux=A[4];
A[4]=A[8];
A[8]=aux;

Adelante


55 18 6 12 94 44 10 8
55 18 6 12 94 44 10 8
55 18 6 12 94 44 10 8
55 18 6 12 94 44 10 8
55 18 6 8 94 44 10 12

Adelante

Se repite este paso (se maneja el mismo intervalo 4 y no hay cambios)



Adelante

Como ya no hay intercambios entre los elementos comparados con intervalo 4, se modifica el intervalo.

4/2, el siguiente tamaño de intervalo es 2

Se reinician las comparaciones con tamaño de intervalo 2

Si A[1] > A[3]

aux = 55;

A[1] = 6;

A[3]= aux;

Si A[2] > A[4]

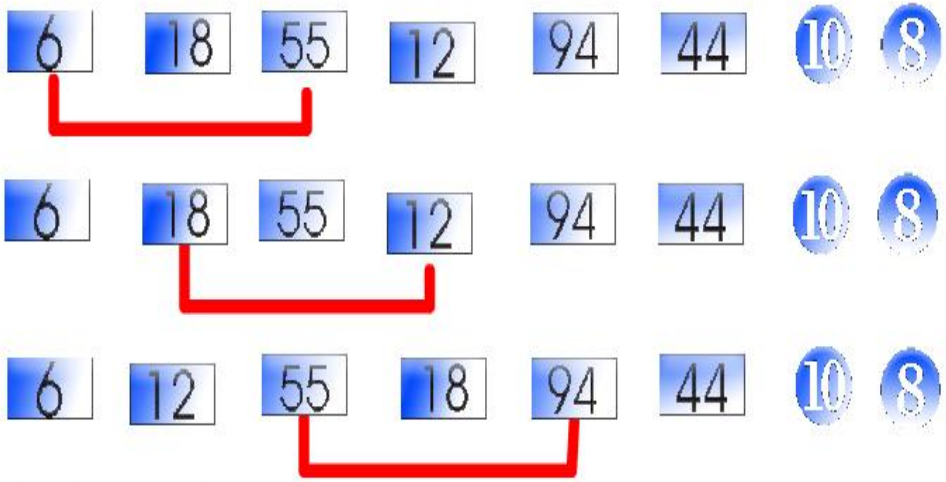
aux = 18;

A[2] = 12;

A[4]= 18;

Adelante

comparaciones con tamaño de intervalo 2

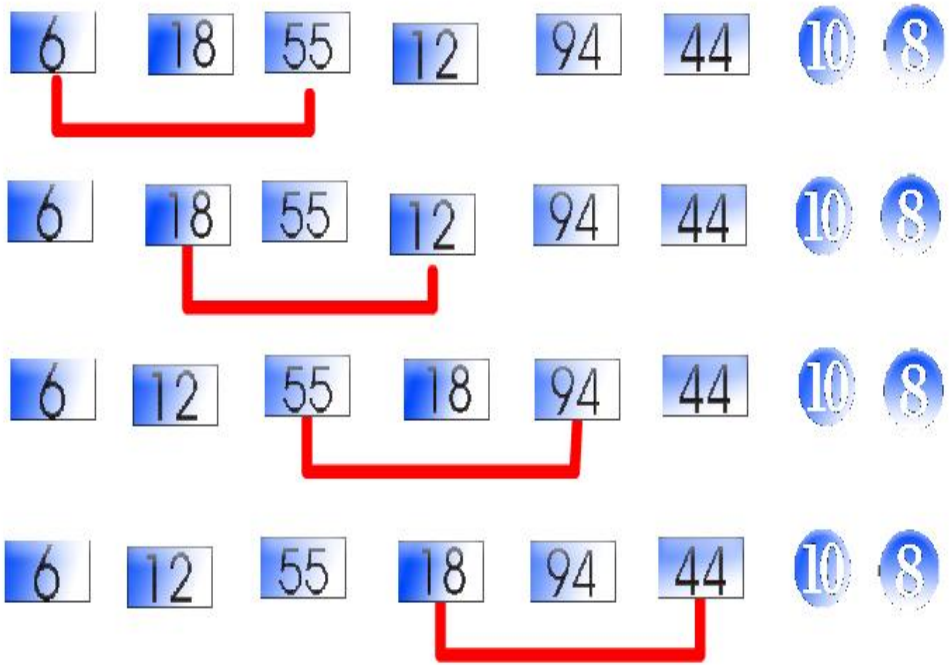


No A[3] > A[5]

No hay intercambio

A[3] = 55;
A[5]= 94;

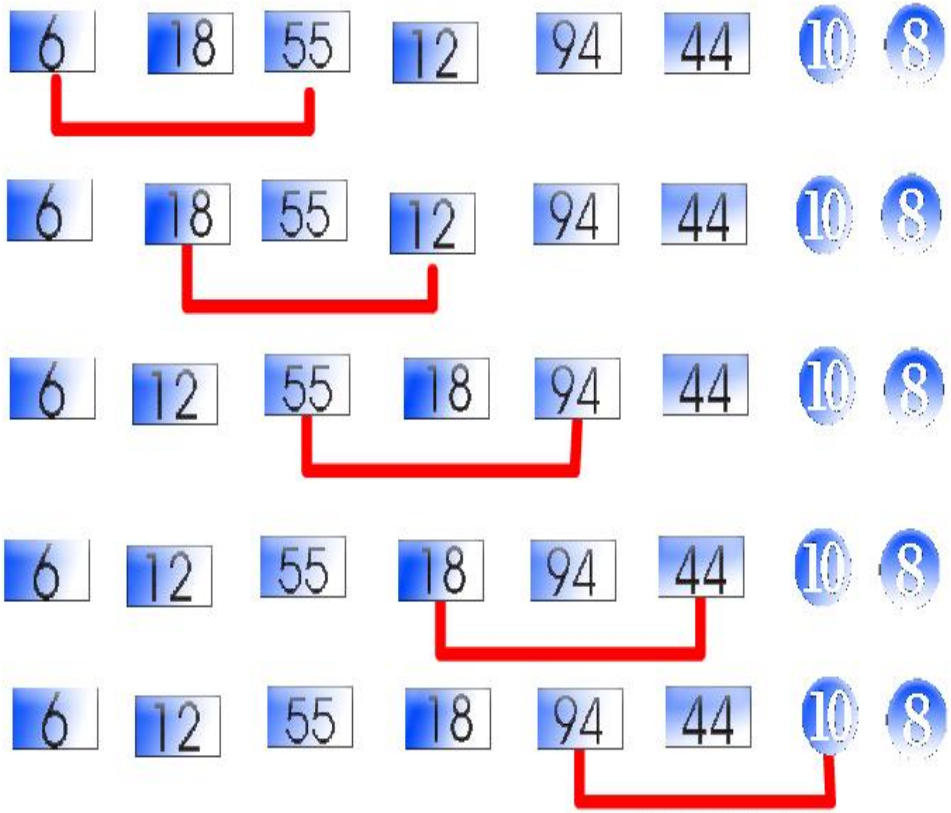
comparaciones con tamaño de intervalo 2



Si A[4] > A[6] Falso;

Adelante

comparaciones con tamaño de intervalo 2



Si A[5] > A[7]

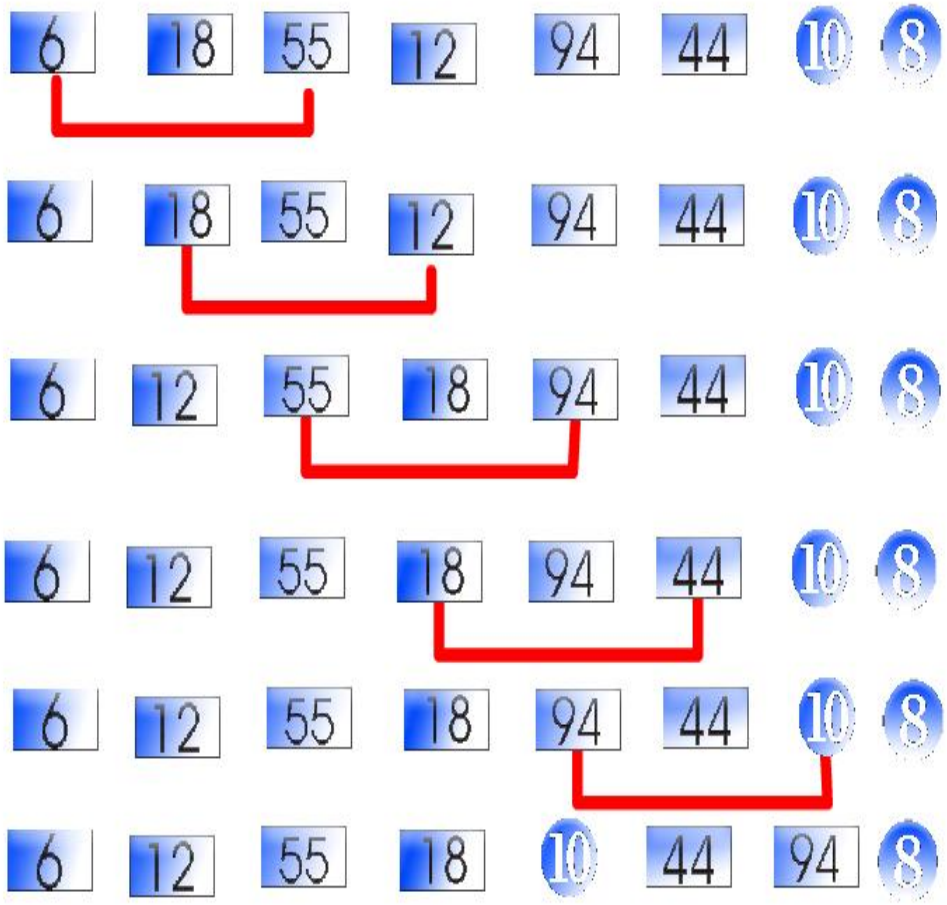
aux = 94;

A[5] = 10;

A[7]= aux;

Adelante

comparaciones con tamaño de intervalo 2



Si A[2]>A[4]
aux=A[2];
A[2]=A[4];
A[4]=aux;

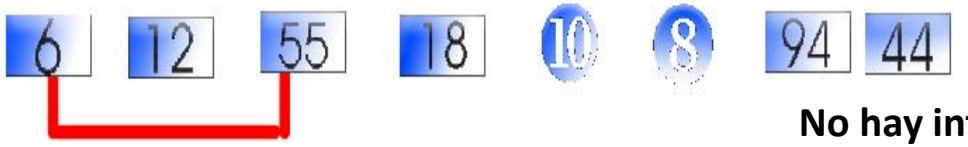
Si A[5]>A[7]
aux=A[5];
A[5]=A[7];
A[7]=aux;

Adelante

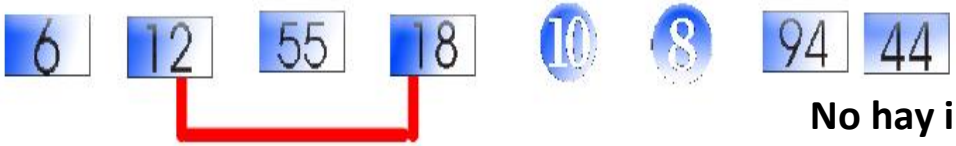
comparaciones con tamaño de intervalo 2

Si A[6] > A[8]
aux = A[6];
A[6] = A[8];
A[8]= aux;

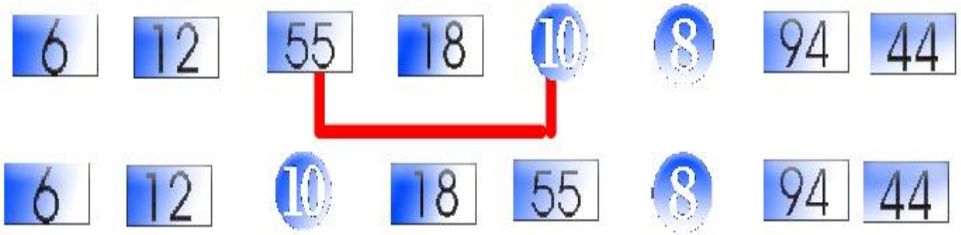
Se vuelven a iniciar las comparaciones con tamaño de intervalo 2



No hay intercambio



No hay intercambio



Si A[3]>A[5]
aux=A[3];
A[3]=A[5];
A[5]=aux;

Adelante

comparaciones con tamaño de intervalo 2

Si A[4] > A[6], intercambia 18 y 8

aux = A[4];
A[4] = A[6];
A[6]= aux;

No hay intercambio

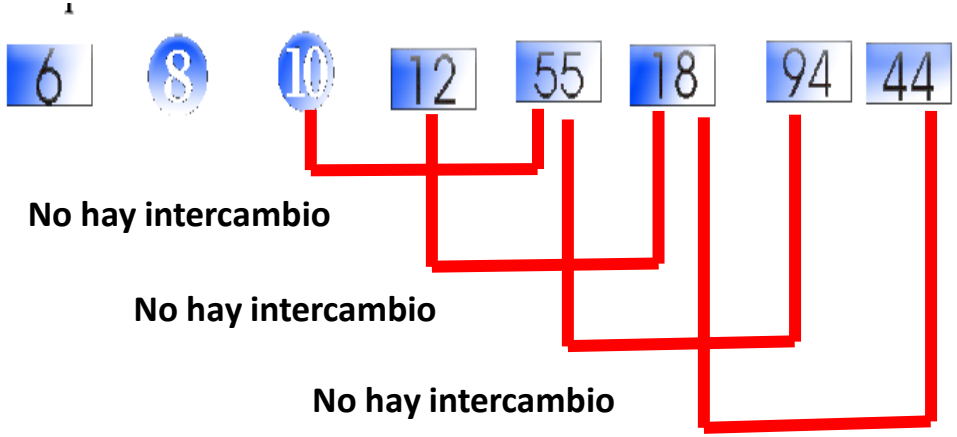
No hay intercambio

Como hubo intercambio, se repite el este paso con el mismo intervalo.

No hay intercambio

Si A[2]>A[4]
aux=A[2];
A[2]=A[4];
A[4]=aux;

Adelante



No hay intercambio

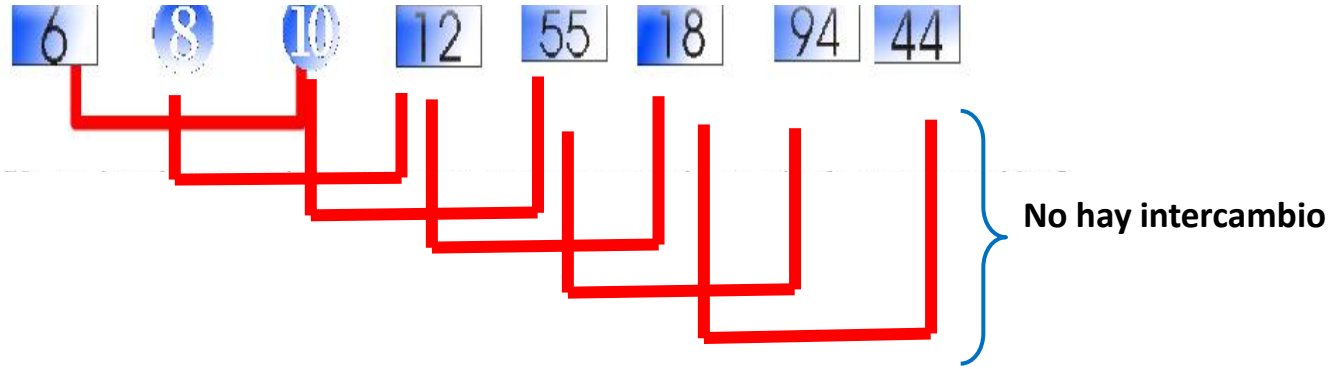
No hay intercambio

No hay intercambio

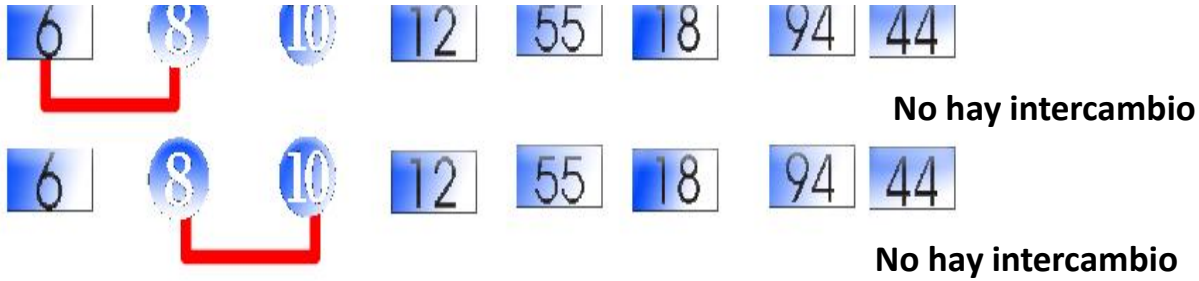
No hay intercambio

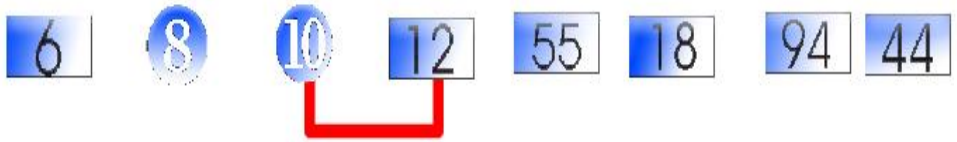
Hubo un intercambio al inicio de esta iteración, se inicia otra pasada con el mismo intervalo

comparaciones con tamaño de intervalo 2

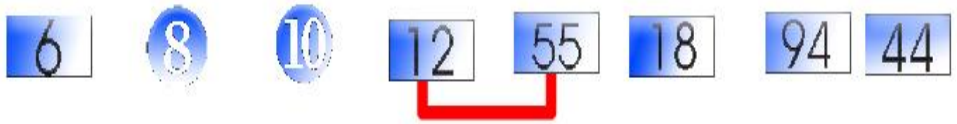


Como ya no hay intercambios, se modifica el intervalo 2/2=1

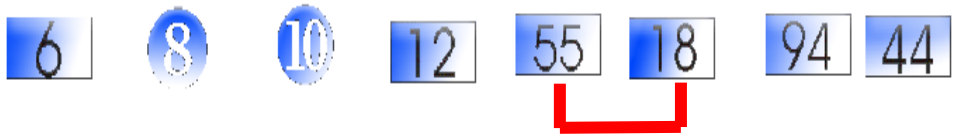




No hay intercambio



No hay intercambio



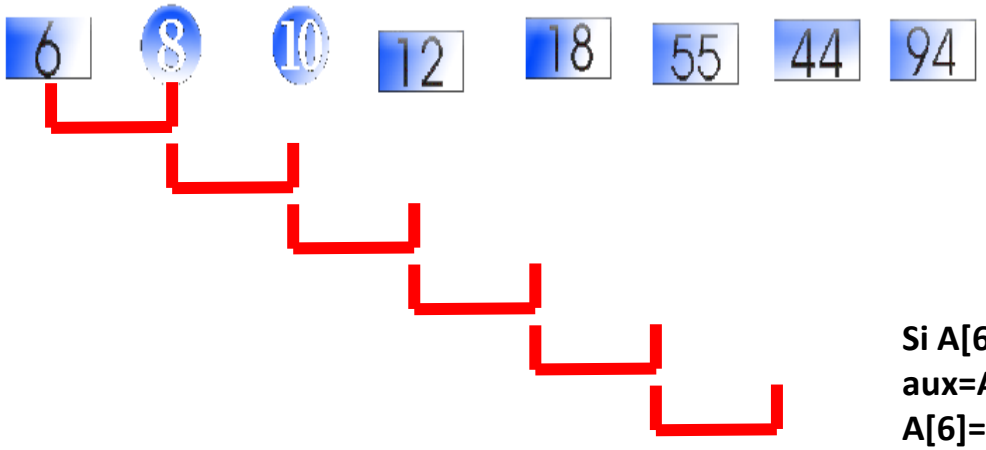
Si A[5]>A[6]
aux=A[5];
A[5]=A[6];
A[6]=aux;

No hay intercambio

Si A[7]>A[8]
aux=A[7];
A[7]=A[8];
A[8]=aux;

Adelante

Se vuelven a iniciar las comparaciones con tamaño de intervalo 1



Si A[6]>A[7]
aux=A[6];
A[6]=A[7];
A[7]=aux;

Hubo un cambio, se repiten comparaciones con el mismo intervalo

No hubo cambios, el arreglo está ordenado

Adelante

ShellSort

código en Java

```
public static void ShellSort ( int [] a, int n )  {
        int I, inter = n ;
        int aux;
        boolean band = true;
        while (inter  > 1 ) {
           inter = inter  / 2;
           band = true;
           while ( band ) {
              band = false;
              I = 0;
              while ( (I + inter ) < n ) {
                 if ( a[I] > a[I+ inter] ) {
                    aux = a[I];
                    a[I] = a[I + inter];
                    a[I + inter] = aux;
                    band = true;
                 }
                 ++I;
              }
           }
        }
}
```

Análisis de eficiencia del método de Shell

El análisis de eficiencia del método de Shell es un problema muy complicado y aún no resuelto. Hasta el momento no se ha podido establecer la mejor secuencia de incrementos cuando n es grande. Cabe recordar que cada vez que se propone una secuencia de intervalos, es necesario *correr* el algoritmo para analizar su tiempo de ejecución.

En 1969, Pratt descubrió que el tiempo de ejecución del algoritmo es del orden de $n * (\log n)^2$.

Videos

SHELL SORT

https://youtu.be/CmPA7zE8mx0?t=3

https://youtu.be/MHW-QNd6IUE?t=5