



# Organizaron de Datos

## Organización Secuencial

Es la organización más simple y la primera en aparecer. La única cuando los únicos dispositivos de almacenamiento permanente eran las cintas magnéticas.

Se basa en el acceso secuencial pero también pueden optimizarse utilizando acceso relativo.

La organización de registros varía según sean de longitud fija o Variable.

Los Registros de longitud Variable actualizables se almacenan en bloques.

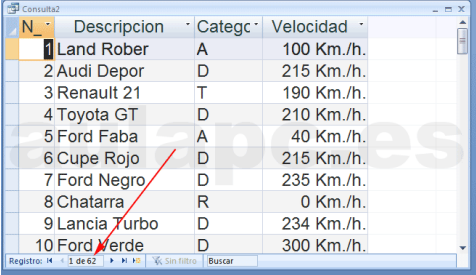
Las primitivas varían según los registros están ordenados o desordenados y según se requiera actualizarlos o no.

En muchos archivos transaccionales sólo se hacen altas al final del archivo (Y nunca se actualizan registros). Ejemplo Logs Bitácoras, etc



# Organizaron Secuencial

## Registros Ordenados por Identificador



N	Descripcion	Categr	Velocidad
1	Land Rober	A	100 Km./h.
2	Audi Depor	D	215 Km./h.
3	Renault 21	T	190 Km./h.
4	Toyota GT	D	210 Km./h.
5	Ford Faba	A	40 Km./h.
6	Cupe Rojo	D	215 Km./h.
7	Ford Negro	D	235 Km./h.
8	Chatarra	R	0 Km./h.
9	Lancia Turbo	D	234 Km./h.
10	Ford Verde	D	300 Km./h.

### Ventajas:

Optimizar búsquedas:

Elimina necesidad de leer todo el archivo

Procesamiento coordinado con otros archivos (**Apareo**)

Cortes de control con un único recorrido (**Reportes**)

### Desventajas:

Problemas de inserción: altas implican nuevo del archivo o deben diferirse.

Bajas costosas: bajas lógicas con necesidad de re-estructuración

*Bajas físicas con reconstrucción del archivo o diferidas*

### Ejemplos:



# Organizaron Secuencial

## Registros Desordenados

### Creación:



Creación y carga sin validación de unicidad ni búsqueda de espacio libre

### Agregar al final:

Apertura de archivo y posicionamiento al final para escribir (**append**).

### Recuperación de Registros:

Posicionamiento al inicio del archivo y lectura secuencial de registros hasta encontrar el buscado o llegar al final del archivo.  
Solo búsqueda lineal.



# Organizaron Secuencial

## Registros Desordenados

### Actualización de Registros:

#### Inserción:

Se busca si el registro existe y si no existe se graba el nuevo buscando el espacio libre

#### Modificación:

Se recupera el registro Buscado, se actualiza, se vuelve al comienzo de la unidad recién leída (registro o bloque) y escribe la unidad modificada.

#### Eliminación: (en RRLF)

Se Busca el registro a eliminar; se toma su posición; se graba el ultimo Registro del archivo en la posición a eliminar y trunca el archivo.

#### Eliminación: (en RRLV)

Se compacta el bloque donde se encontró el registro; se actualiza el Espacio Libre del bloque y se reescribe el bloque



# Organizaron Secuencial

## Registros Ordenados

### Manipulación de Registros:

#### Creación:

Registros a cargar ya están validados y ordenados por identificador ordenamiento externo de un archivo desordenado

#### Recuperación de Registros:

Consulta o recuperación unitaria. [Búsqueda Lineal o Binaria.](#)

#### Reportes:

[cortes de control](#) (Registros ordenados por mas de un campo. Se puede agrupar, Con Totales y subtotales, contar, sumar, promediar, etc.))



# Organizaron Secuencial

## Registros Ordenados

### Actualización de Registros:

#### Inserción: (Directa)

Hay que crear un archivo nuevo, copiar los registros con identificadores menores al del registro a insertar, agregar el registro nuevo, copiar el resto de los registros con identificadores mayores, borrar el archivo viejo y renombrar el archivo nuevo con el nombre del viejo

#### Inserción: (Por novedades)

Crear un nuevo archivo con las altas ordenadas; luego insertar por mezcla. Ojo: Recuperar un Registro-consultar el archivo principal y el de novedades

#### Modificación:

Posiciona el registro por el identificador, se desde el comienzo de la unidad all registro y se Escribe.

Otra Manera es hacerlo por Apareo Maestro-Novedades-Nuevo Maestro



# Organizaron Secuencial

## Registros Ordenados

### Eliminación: (Directa)

Hay que crear un archivo nuevo, copiar los registros con identificadores menores al del registro a Eliminar, copiar el resto de los registros con identificadores mayores, borrar el archivo viejo y renombrar el archivo nuevo con el nombre del viejo

### Eliminación: (Por novedades)

Crear un nuevo archivo con Los Registros a Eliminar; **luego Eliminar por mezcla.**

### Eliminación: (Por marcado)

- a) Ubicar el registro por el identificador, Cambiar signo del Identificador
- b) Ubicar el registro por el identificador, campo de marcado (costo x bits)

**Mantenimiento:** Hacer siempre copia de respaldo. En al caso de Eliminación por marcado. Hay que cada tanto depurar o compactar.



# Organizaron Secuencial

## Registros Ordenados

### Eliminación: (Directa)

Hay que crear un archivo nuevo, copiar los registros con identificadores menores al del registro a Eliminar, copiar el resto de los registros con identificadores mayores, borrar el archivo viejo y renombrar el archivo nuevo con el nombre del viejo

### Eliminación: (Por novedades)

Crear un nuevo archivo con Los Registros a Eliminar; luego Eliminar por mezcla.

### Eliminación: (Por marcado)

- a) Ubicar el registro por el identificador, Cambiar signo del Identificador
- b) Ubicar el registro por el identificador, campo de marcado (costo x bits)

**Mantenimiento:** Hacer siempre copia de respaldo. En al caso de Eliminación por marcado. Hay que cada tanto depurar o compactar.





# Organizaron Secuencial

## Corte de Control

### **Definición:**

Es un proceso en el cual partiendo de registros ordenados por uno o más campos, se los procesa en categorías determinadas por los criterios de ordenamiento.

En otras palabras, se procesa un conjunto ordenado de registros en subconjuntos determinados por los criterios de orden.

### **Aplicaciones del Corte de Control:**

La aplicación más común para la cuál se realiza el corte de control es para generar reportes que acumulen cantidades.

Se utilizan contadores, Sumadores, Promedios.

Se informan SubTotales por Niveles y Totales Generales



# Organizaron Secuencial

## COMIENZO CORTE DE CONTROL

**mientras (no sea fin de Archivo)**

anterior1=Clave1

[inicializar contadores para corte1]

**mientras(no sea fin de Archivo Y anterior1 = clave1)**

**anterior2 = clave2**

**[inicializar contadores para corte2]**

**mientras(no sea fin de Archivo Y anterior1 = clave1 Y anterior2 =clave2)**

**[incrementar contadores corte2]**

**[escribir detalle del corte2]**

**[Leer Archivo]**

**[escribir totales acumulados del corte2]**

**[incrementar contadores del corte1]**

**[escribir totales acumulados del corte1]**

**[incrementar contadores TOTALES GENERALES]**

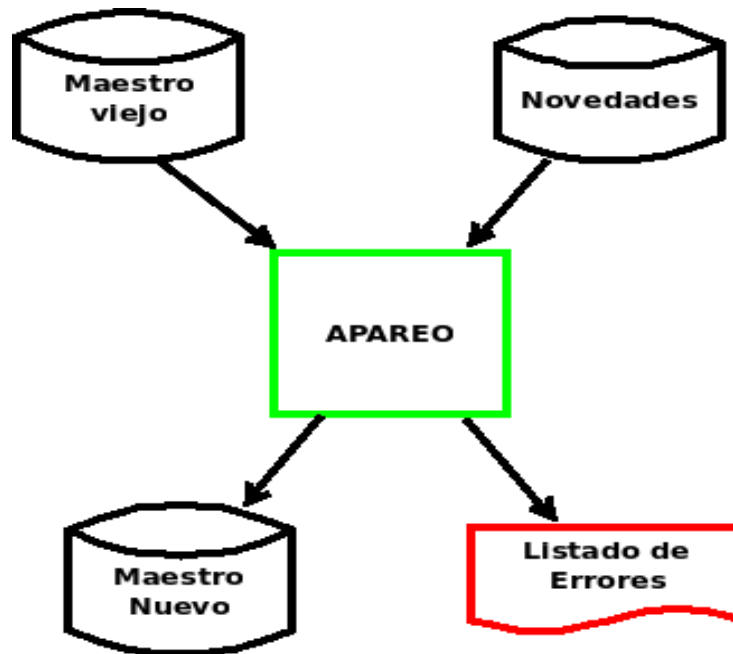
**ESCRIBIR TOTALES GENERALES**



# Organizaron de Datos

## Apareo de Archivos

A partir de dos archivos (uno principal y otro relacionado) y tienen alguna información que los enlace, generar un tercero (o una salida por pantalla), como una mezcla de los dos.



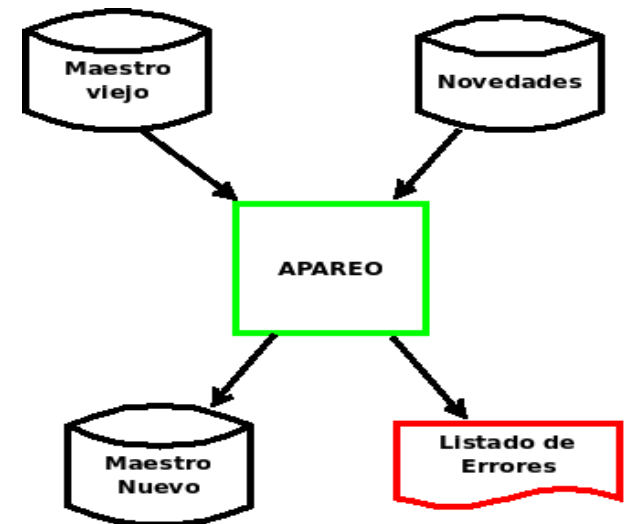
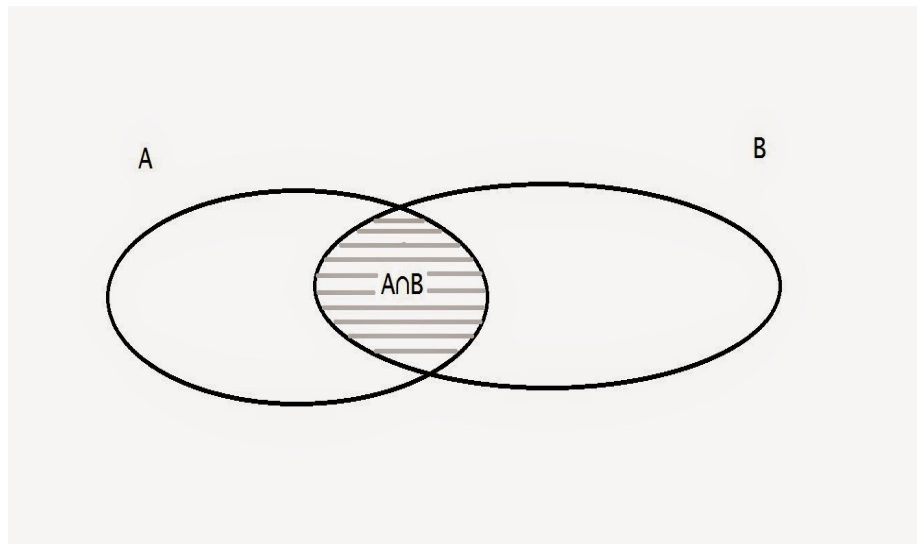


# Organizaron de Datos

## Apareo de Archivos

### Intersección:

A partir de dos archivos ambos ordenados (relacionado por sus identificadores)



**Usos:** Quedarme con los comunes

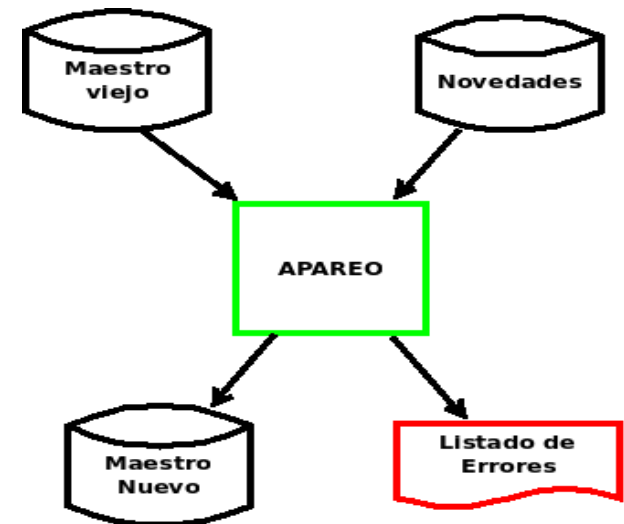
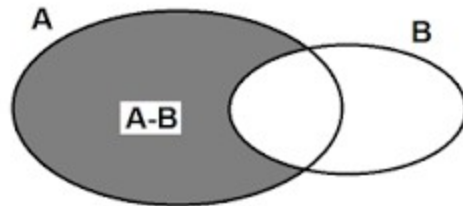


# Organizaron de Datos

## Apareo de Archivos

### Diferencia:

A partir de dos archivos amos ordenados  
(relacionado por sus identificadores)



**Usos:** Quedarme solo con los del Maestro-Novedades.  
Bajas

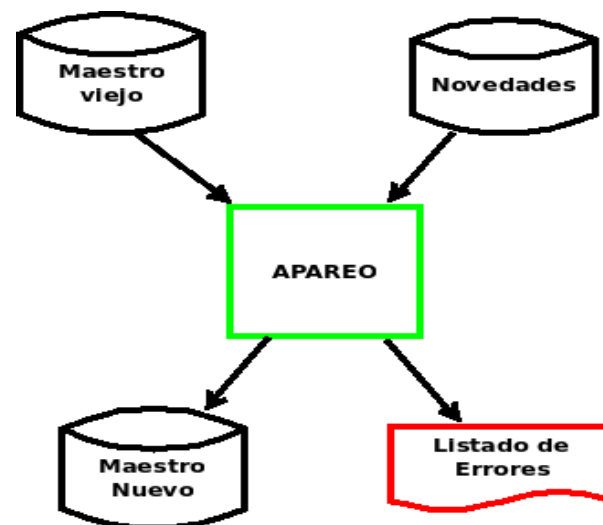
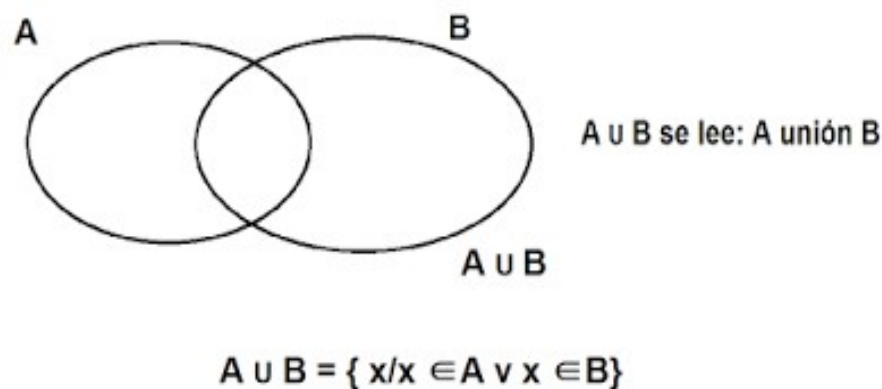


# Organizaron de Datos

## Apareo de Archivos

### Unión:

A partir de dos archivos amos ordenados  
(relacionado por sus identificadores)



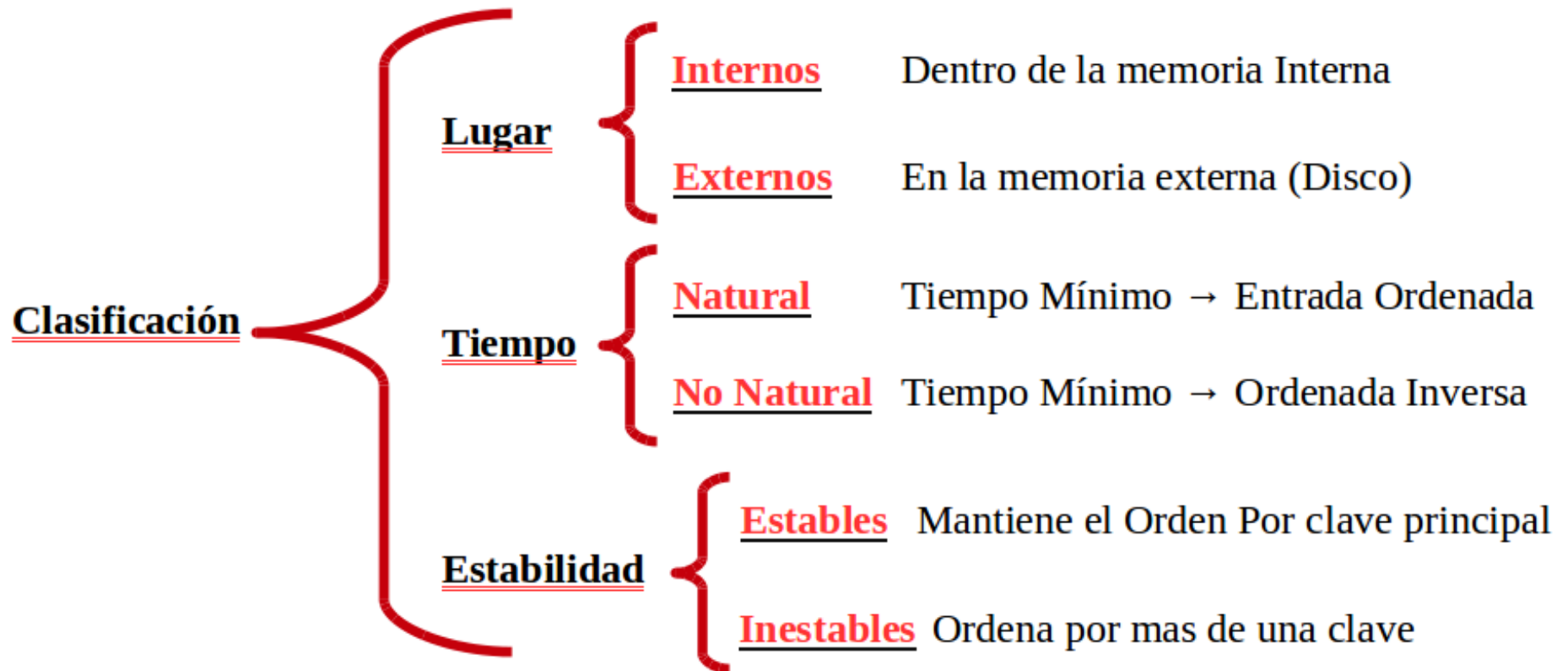
**Usos:** Quedarme solo con los del Maestro + Novedades.  
Implica altas e inserciones.  
Obtengo el mezclado de los Archivos.



# Organizaron de Datos

## Ordenamiento de Archivos

Los Algoritmos de Búsqueda Binaria, Fusión o apareo, Corte de Control, etc. Requieren de tener ordenados los datos.



Nomenclatura:



# Ordenamiento de Archivos

## Ordenamiento de Archivos

### Comparación de Algunos Métodos de Ordenamiento:

Algoritmo	Nombre	Complejidad	Memoria	Método	Estable
Burbuja	Bubblesort	$O(n^2)$	$O(1)$	Intercambio	Si
Inserción	Insertion sort	$O(n^2)$ "o menor"	$O(1)$	Inserción	Si
Urnas	Bucket sort	$O(n)$	$O(n)$		Si
Mezcla	Merge sort	$O(n \log n)$	$O(n)$	Mezcla	Si
Shell	Shell sort	$O(n^{1.25})$	$O(1)$	Inserción	No
Selección	Selection sort	$O(n^2)$	$O(1)$	Selección	No
Monticulos	Heapsort	$O(n \log n)$	$O(1)$	Selección	No
Rapido	Quick Sort	$O(n \log n)$ , $O(n^2)$	$O(\log n)$	Dividir	No

Complejidad Computacional=  $O(n)$  y Uso de Memoria=  $O(n)$





# Ordenamiento de Archivos

## BURBUJA

**Evalúa cada elemento de la lista con el siguiente intercambiándolos de posición si están en el orden equivocado**

COMIENZO

PARA  $i=1$  HASTA  $N$  HACER

PARA  $J=1$  HASTA  $N-1$  HACER

SI  $\text{dato}[j] > \text{dato}[j+1]$  ENTONCES

$\text{aux} = \text{datos}[J]$

$\text{dato}[j] = \text{dato}[j+1]$

$\text{dato}[j+1] = \text{aux}$

FIN\_SI

FIN\_PARA

FIN\_PARA

FIN

COMIENZO

REPETIR

ordenado = FALSE

PARA  $i=1$  HASTA  $N-1$  HACER

SI  $\text{dato}[i] > \text{dato}[i+1]$  ENTONCES

$\text{aux} = \text{datos}[i]$

$\text{dato}[i] = \text{dato}[i+1]$

$\text{dato}[i+1] = \text{aux}$

ordenado = FALSE

FIN\_SI

FIN\_PARA

Hasta ordenado = TRUE

FIN



# Ordenamiento de Archivos

## INSERCIÓN

**Divide la lista en Parte Ordenada y Desordenada.  
Evalúa cada elemento de a Lista Desordenada  
y lo inserta en la lista ordenada de a uno.**

COMIENZO

PARA  $i=2$  HASTA  $N$  HACER

    elemento = dato[ $i$ ]

$j = i - 1$

    MIENTRAS  $j \geq 1$  Y dato[ $j$ ] > elemento HACER

        dato[ $j+1$ ] = dato[ $j$ ]

$j = j - 1$

    FIN\_MIENTRAS

    dato[ $j+1$ ] = elemento

FIN\_PARA

FIN

45	17	23	67	21
----	----	----	----	----

Iteración 1:

Se ordenan los primeros dos elementos.

17	45	23	67	21
----	----	----	----	----

Iteración 2:

Se ordenan los elementos 3 y 4.

17	23	45	67	21
----	----	----	----	----

Se toma el tercer elemento 23 y se ordena entre el primer elemento y el segundo, recorriéndose el 45 a la tercera posición.

Iteración 3:

17	23	45	67	21
----	----	----	----	----

No hay cambio ya que el 67 es mayor al 45.

Iteración 4:

17	23	45	67	21
----	----	----	----	----

Se toma el número 21, y se mete en la posición correspondiente del vector, recorriendo los números mayores.

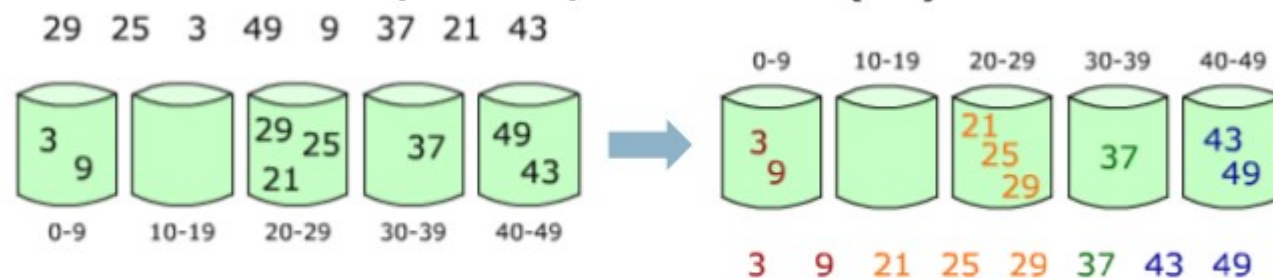
17	21	23	45	67
----	----	----	----	----



# Ordenamiento de Archivos

## Ordenamiento x Urnas

Se divide el array en un conjunto de “urnas” y cada urna se ordena por separado:  $O(n^2)$



Bajo ciertas condiciones (claves entre 0 y  $N-1$  sin duplicados), la ordenación se puede hacer en  $O(n)$ :

Algoritmo:

- Crear una colección de casilleros vacíos
- Colocar cada elemento a ordenar en un único casillero
- Ordenar individualmente cada casillero
- devolver los elementos de cada casillero concatenados por orden



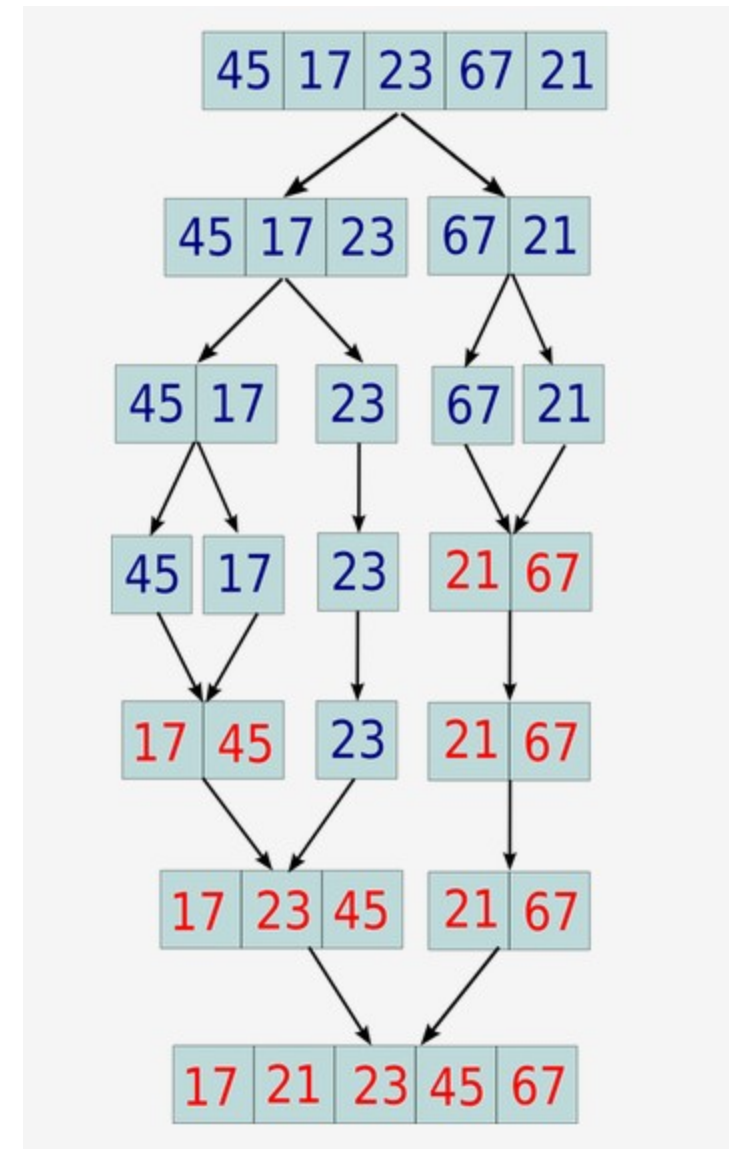
# Ordenamiento de Archivos

## Ordenamiento x Mezcla

1. Si la longitud de la lista es 0 ó 1, entonces ya está ordenada.

En otro caso:

2. Dividir la lista desordenada en dos sublistas de aproximadamente la mitad del tamaño.
3. Ordenar cada sublista recursivamente aplicando el ordenamiento por mezcla.
4. Mezclar las dos sublistas en una sola lista ordenada.





# Ordenamiento de Archivos

## Ordenamiento x Mezcla

```
FUNCION Ordena_MergeSort(vec)
COMIENZO
  N = vec.length
  SI N <= 1 ENTONCES
    RETORNA vec
  SINO
    mitad= N/2
    CopiaVector(izq,1,mitad)
    copiaVector(der,mitad+1,N)
    Ordena_MergeSort(izq)
    Ordena_MergeSort(der)
    mezcla(vec, izq, der);
  FIN_END
FIN {Funcion}
```

```
FUNCION Mezcla(vec, izq, der)
COMIENZO
  N = vec.length
  M = izq.length
  O = der.length
  i = 1 j = 1
  PARA k =1 HASTA N HACER
    SI i >= M ENTONCES
      vec[k] = der[j]
      j = j + 1
    SINO SI j >= O ENTONCES
      vec[k] = izq[i]
      i = i + 1
    SINO
      vec[k] = der[j]
      j = j + 1
  FIN_END
FIN_PARA
FIN
```



# Ordenamiento de Archivos

## Shell Sort

Este algoritmo de ordenación fue creado por Donald Shell.  
**Mejora al ordenamiento por inserción comparando elementos separados por un espacio de varias posiciones.**

Generalmente se toma como intervalo inicial  $n \div 2$ ,  
Luego se reduce los intervalos a la mitad hasta  
que el intervalo llegue a ser uno y aplica inserción.

**Existen Varias Variantes de este Algoritmo:**

- ▮ Aplicando Intervalos de  $N \text{ DIV } 2$
- ▮ Aplicando Gonnet- Baeza-Yates  $N * 5 \text{ DIV } 11$ .
- ▮ Aplicando Steven Pigeon

$n-2$

Secuencia matemática:  $a_n = 1 + e$        $e \rightarrow \text{Euler}$





# Ordenamiento de Archivos

## Shell Sort

Generalmente se toma como intervalo inicial  $n \div 2$ ,

```
PROCEDURE Ordenar(A:LNaturales);
VAR i,j:longint;
    aux:qword;
    intv:longword; //intervalo
BEGIN
    intv:=length(A) DIV 2; //intervalo inicial

    WHILE intv>0 DO // Mientras intervalo >0
        BEGIN

            //algoritmo de inserción, por intervalos

            FOR i:=intv TO high(A) DO // Desde el intervalo hasta el final
                BEGIN
                    aux:=A[i];
                    j:=i-intv;
                    WHILE ((j>=0) AND (aux<A[j])) DO
                        BEGIN
                            A[j+intv]:=A[j];
                            j:=j-intv
                        END;
                    A[j+intv]:=aux
                END;

                //nuevo intervalo

                intv:=intv DIV 2
            END;
        END;
    END;
```



# Ordenamiento de Archivos

## Shell Sort (Mejora x Gonnet y Baeza-Yates)

La secuencia de intervalos consiste en tomar  $N * 5 \text{ DIV } 11$  para el siguiente Valor, Secuencia  $*5 \text{ DIV } 11$  (Sucesivamente)  
Si Valor=1 cambiar a Valor=2

```
PROCEDURE Ordenar(A:LNaturales);  
  
VAR i,j:longint;  
    aux:qword;  
    intv:longword;    // intervalo  
  
BEGIN  
    intv:=(length(A)*5) DIV 11;    //intervalo inicial  
    WHILE intv>0 DO    // Mientras intervalo > 0  
        BEGIN  
            //algoritmo de inserción, por intervalos  
  
            FOR i:=intv TO high(A) DO // Desde el intervalo hasta el final  
                BEGIN  
                    j:=i;  
                    aux:=A[i];  
                    WHILE ((j>=intv) AND (auxA<[j-intv])) DO  
                        BEGIN  
                            A[j]:=A[j-intv];  
                            j:=j-intv  
                        END;  
                    A[j]:=aux  
                END;  
  
                // nuevo intervalo  
  
                intv:=(intv * 5) DIV 11 ;  
                IF intv=2 THEN intv:=1  
            END;  
        END;  
    END;
```





# Ordenamiento de Archivos

## SELECT SORT

Recorre todo el arreglo desde la primera posición, buscando el menor elemento de todos. Pone el Menor en la Primera Posición. Repite todo el proceso para la sig posición y así sucesivamente.

6	5	20	1	3	8	4	10	40	12
1	5	20	6	3	8	4	10	40	12
1	3	20	6	5	8	4	10	40	12
1	3	4	6	5	8	20	10	40	12
1	3	4	5	6	8	20	10	40	12
1	3	4	5	6	8	20	10	40	12
1	3	4	5	6	8	20	10	40	12
1	3	4	5	6	8	10	20	40	12
1	3	4	5	6	8	10	12	40	20
1	3	4	5	6	8	10	12	20	40



# Ordenamiento de Archivos

## QUICK SORT

La regla de este Algoritmo es Divide y triunfaras... Permite, en promedio, ordenar  $n$  elementos en un tiempo proporcional a  $n \log n$ . Esta es la técnica de ordenamiento más rápida conocida.

Unsorted Array



### PASOS:

- 1) Elegir un elemento de la lista, al que llamaremos pivote.
- 2) Situar los demás de la lista a cada lado del pivote, de forma que a un lado queden todos los menores que él, y al otro los mayores.
- 3) Cuando los Índices se junten: Resituar el Pivote.
- 4) Repetir este proceso de forma recursiva para cada sublista mientras éstas contengan más de un elemento.



# Ordenamiento de Archivos

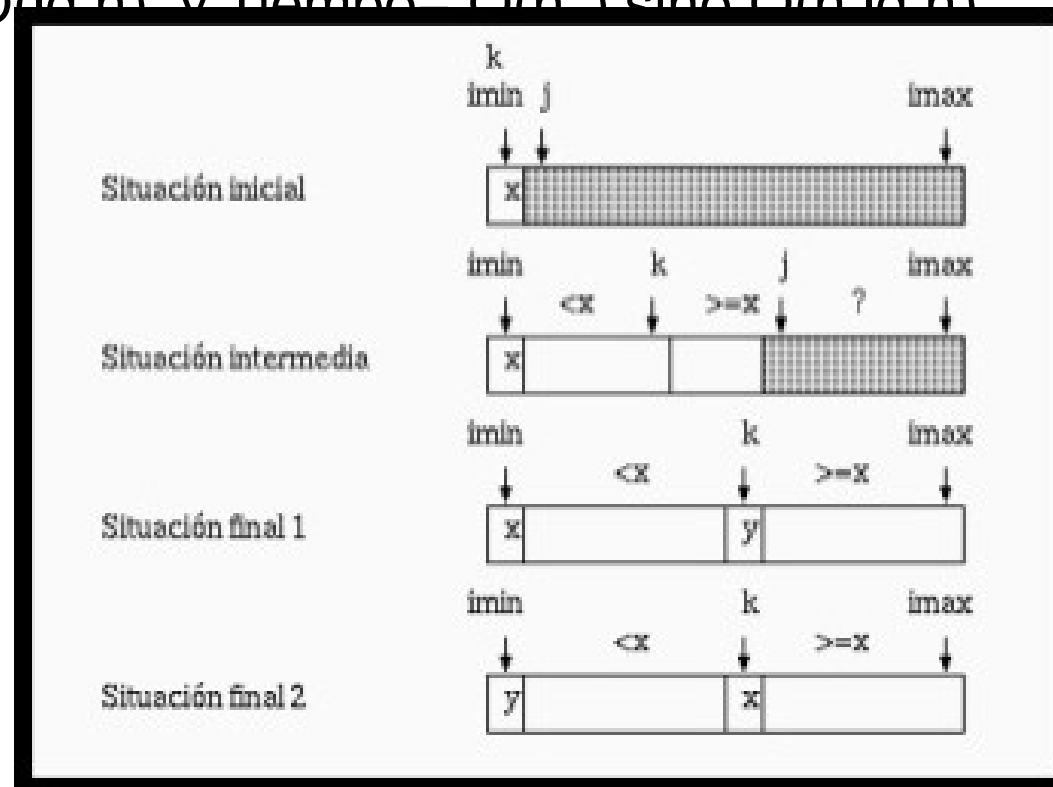
## QUICK SORT

Propiedades de este Algoritmo:

Es Inestable. No se Adapta y es recursivo

2

Complejidad  $O(\lg n)$  y Tiempo =  $O(n)$  sino  $O(n \lg n)$





# Ordenamiento de Archivos

## Ordenamiento en Disco

El Problema: “La memoria principal no alcanza”.

El ordenamiento externo se requiere cuando la información a procesar no cabe en la memoria principal de una computadora (RAM).

Hay que utilizar Memoria externa ( un disco). El proceso es mas lento.

- ¿ Cuanto espacio necesito?
- ¿ Sobre-escribo el Archivo de Origen?
- ¿ Me sirve cualquier método?
- ¿ Cuanta RAM puedo utilizar?



Los métodos de ordenamiento externos más comunes:

Mezcla Directa (o Merge Sort)

Mezcla Natural (o Natural Merge Sort).



# Ordenamiento de Archivos

## Mezcla Directa (Merge Sort)

El Problema: “La memoria principal no alcanza”. Utilizo lo que puedo.

Ejemplo:

Problema Ordenar 900 MB utilizando únicamente 100 MB de RAM.

- 1) leer 100MB de información en RAM y ORDEANAR.
- 2) Escribase la información ordenada en el disco.  
Repítanse los pasos 1 y 2 hasta el Final.
- 3) Ahora se deben mezclar todos los pedazos ordenados:  
Generar Buffer de Salida de (10 Mb)  
Leer los primeros 10 MB de cada pedazo (90 Mb).  
Ordenar los 9 pedazos mezclándolos y  
grábese el resultado en el buffer de salida.  
Si el buffer de salida está lleno  
    escribase al archivo destino final  
Si cualquiera de los 9 buffers leídos queda vacío  
    se llena con los siguientes 10 MB de su pedazo de 100 MB  
    o se marca este como completado si ya no hay registros remanentes.



# Ordenamiento de Archivos

## Mezcla Natural (Natural Merge Sort)

El Problema: “La memoria principal no alcanza”. **Utilizo lo que puedo.**

El método consiste en aprovechar la existencia de secuencias ya ordenadas dentro de los datos de los archivos.

### Pasos:

- 1) A partir de las secuencias ordenadas existentes en el archivo, se obtienen particiones que se almacenan en dos archivos auxiliares.
- 2) Las particiones almacenadas en estos archivos auxiliares se Aparean posteriormente para crear secuencias ordenadas.
- 3) **1 y 2** Se repiten hasta arbitrariamente hasta lograr ordenación de los datos contenidos en el archivo original.





# Ordenamiento de Archivos

## Mezcla Natural (Natural Merge Sort)

Un ejemplo de Ordenamiento:

1º Interacción:

Archivo ORIGINAL

10	17	05	08	09	03	22	50	04	02	20	30	11	16	19	06	21	23
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

Archivo AUXILIAR1

10	17	03	22	50	02	20	30	06	21	23
----	----	----	----	----	----	----	----	----	----	----

Archivo AUXILIAR2

05	08	09	04	11	16	19
----	----	----	----	----	----	----

Archivo ORDENADO

05	08	09	10	17	03	04	11	16	19	22	50	02	20	30	06	21	23
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----



# Ordenamiento de Archivos

## Mezcla Natural (Natural Merge Sort)

### Un ejemplo de Ordenamiento: (Cont)

2º Interacción:

Archivo ORDENADO

05	08	09	10	17	03	04	11	16	19	22	50	02	20	30	06	21	23
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

Archivo AUXILIAR1

05	08	09	10	17	02	20	30
----	----	----	----	----	----	----	----

Archivo AUXILIAR2

03	04	11	16	19	22	50	06	21	23
----	----	----	----	----	----	----	----	----	----

Archivo ORDENADO

03	04	05	08	09	10	11	16	17	19	22	50	02	06	20	21	23	30
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----





# Ordenamiento de Archivos

## Mezcla Natural (Natural Merge Sort)

Un ejemplo de Ordenamiento: (cont.)

3º Interacción:

Archivo ORDENADO

03	04	05	08	09	10	11	16	17	19	22	50	02	06	20	21	23	30
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

Archivo AUXILIAR1

03	04	05	08	09	10	11	16	17	19	22	50
----	----	----	----	----	----	----	----	----	----	----	----

Archivo AUXILIAR2

02	06	20	21	23	30
----	----	----	----	----	----

Archivo ORDENADO

02	03	04	05	06	08	09	10	11	16	17	19	20	21	22	23	30	50
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----



# Ordenamiento de Archivos

## Mezcla Natural (Natural Merge Sort)

**Conclusión:** Se requieren para procesar, 2 Archivos como Mínimo  
Por seguridad se recomienda uno mas. (3 ARCHIVOS)

**ARCHIVO ORIGEN.**

AUXILIAR1.  
AUXILIAR2.  
ORDENADO ( TEMPORAL).

REPETIR

**ARCHVO ORDENADO.**