Unidad 5 Ordenamiento Externo

Se requiere una cantidad de espacio extra proporcional al tamaño del archivo. Sin embargo, no necesariamente se requiere espacio para todos los registros; dependiendo del método y del algoritmo empleado, solo se requerirá espacio extra para las llaves o para apuntadores a los registros.

7.1 Mezclas

Este grupo de algoritmos toma 2 archivos ordenados y produce un nuevo archivo también ordenado.

7.1.1 Mezcla Simple

arch1 y **arch2** son dos archivos ya ordenados.

Por ejemplo:

arch1	5	10	28	31	55	92	100	131	139

arch2	3	7	8	40	56

arch3 deberá contener todos los registros de arch1 y arch2 también en orden.

¿Qué métodos de los que conocemos podríamos emplear?

Mezcla Simple

arch1	5	10	28	31	55	92	100	131	139
arch2	3	7	8	40	56				

arch3	3													
	3	5												
	3	5	7											
	3	5	7	8										
	3	5	7	8	10									
	3	5	7	8	10	28								
	3	5	7	8	10	28	31							
	3	5	7	8	10	28	31	40						
	3	5	7	8	10	28	31	40	55					
	3	5	7	8	10	28	31	40	55	56				
	3	5	7	8	10	28	31	40	55	56	92	100	131	139
arch3	3	5	7	8	10	28	31	40	55	56	92	100	131	139

Algoritmo:

1. Leer el primer registro de cada archivo (**arch1** y **arch2**)
2. Crear un archivo nuevo vacío y llamarlo **arch3**.
3. Se comparan las llaves de **arch1** y **arch2**.
    4. El registro con la llave menor de los dos se guardará en **arch3**.
    5. Se toma el siguiente registro de **arch1** o **arch2** (el que corresponda con el último que se escribió en **arch3**).
    6. Ir al paso 3.
7. Cuando uno de los archivos llegue al final, los registros restantes del otro se traspasarán a **arch3**.

Pregunta: ¿Cuál sería el algoritmo para mezclar **"m"** archivos ordenados?

Ventaja: Orden de complejidad **temporal O(n)**
El tiempo de ejecución depende directamente del tamaño del archivo.

Desventajas: Orden de complejidad **espacial O(n).** Se requiere una cantidad de espacio extra proporcional al tamaño de ambos archivos.
Este método solo se puede aplicar si los archivos origen están previamente ordenados.

7.1.2 Intercalación o Mezcla Directa.

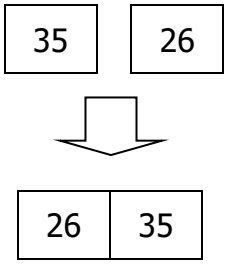
Fundamentos:

- De acuerdo al algoritmo de Mezcla Simple sabemos que se puede obtener un archivo ordenado a partir de dos previamente ordenados.
- Los archivos de tamaño **UNO** ya están ordenados.

Llaves de un archivo al azar:

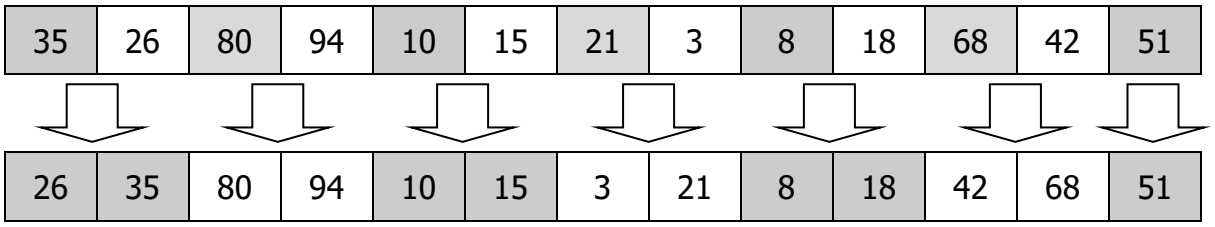
35	26	80	94	10	15	21	3	8	18	68	42	51

Tomemos las 2 primeras llaves del archivo y **veámoslas** como 2 archivos independientes **de tamaño 1 (por lo tanto ya ordenados)**, si los mezcláramos para formar uno solo, el resultado sería:



Primera Etapa:

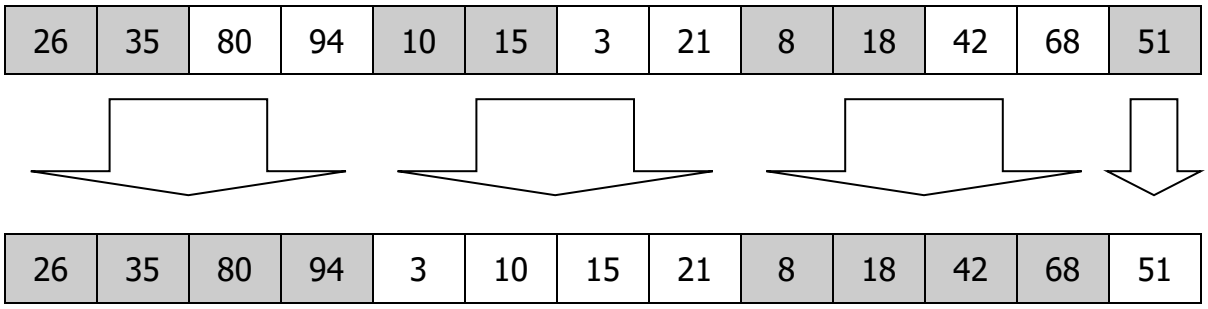
La intercalación directa nos dice que inicialmente hagamos mezclas simples en parejas de archivos de tamaño 1:



Observe como la última llave no se mezcla con ninguna otra.

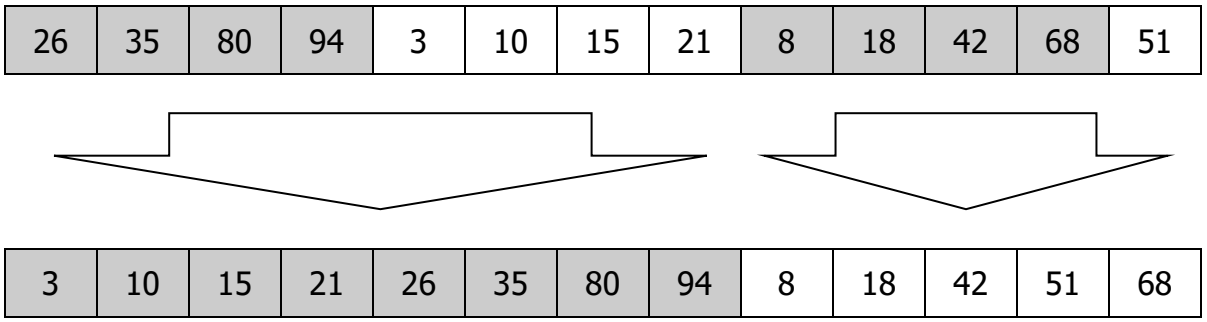
Segunda Etapa:

El archivo ahora se conforma de parejas de llaves ordenadas, por lo que se puede hacer una mezcla de nuevo.



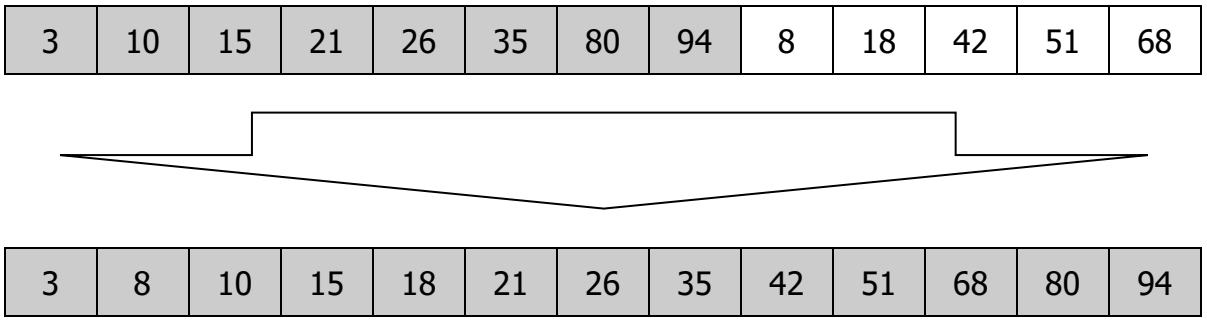
Tercera Etapa:

El archivo ahora se conforma de grupos ordenados de 4 llaves c/u, por lo que se puede hacer una mezcla de nuevo. Observe que hay un grupo de 4 llaves que se mezclará con un grupo de 1 llave.



Cuarta Etapa:

El archivo ahora se conforma de grupos ordenados de 8 llaves c/u (aunque, en este caso, al ser **N** pequeño, el otro único grupo ordenado es de solo 5 llaves). Se realiza la **mezcla final** entre los **dos subarchivos**.



Observemos como, al aplicar, sucesivamente mezclas simples, a grupos de llaves, se puede lograr obtener ordenado un archivo cualquiera.

Los grupos se forman sucesivamente de tamaño $\mathbf{2^m}$, siendo m de un valor mínimo de **0** y máximo tal que $\mathbf{2^m}$ nunca sea mayor que el tamaño del archivo (**N**).

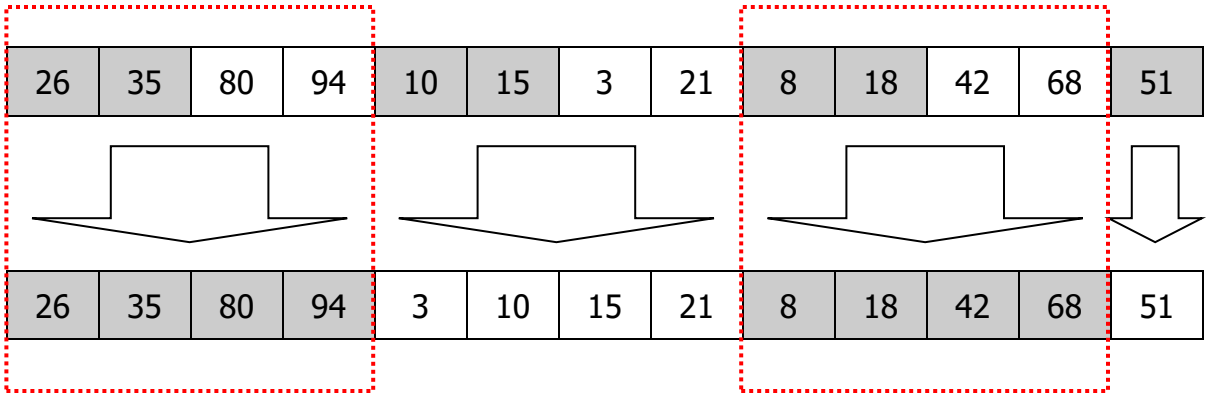
Un algoritmo que use este método de ordenamiento, tiene la ventaja de que su orden de complejidad temporal es **O(n log n)** pero el orden de complejidad espacial es, como todo este tipo de métodos, **O(n)**.

Como vimos en la descripción del algoritmo, no importa que el tamaño del archivo (**N**) no sea una **potencia de 2** (en caso de que lo fuera, lo único que sucede es que todos los subarchivos serían siempre del mismo tamaño).

7.1.3 Mezcla Natural

El archivo original que se usó para el método de intercalación directa es aleatorio, sin embargo, observen como en la segunda etapa, la **primera** y la **tercera parejas de subarchivos** a mezclar, ya conforman de por sí subarchivos **ordenados**.

Esto significa que bajo intercalación directa se hace trabajo innecesario. La mezcla natural resuelve este problema.



Trabajaremos con el mismo archivo que usamos para el método anterior:

35	26	80	94	10	15	21	3	8	18	68	42	51

En la figura siguiente se muestran de la manera acostumbrada (sombreados y no sombreados alternativamente) los subarchivos lógicos de acuerdo a dos conceptos que en inglés se denominan ***runs*** y ***stepdowns***. En castellano, en este documento les llamaremos **subidas** y **bajadas**.

35	26	80	94	10	15	21	3	8	18	68	42	51

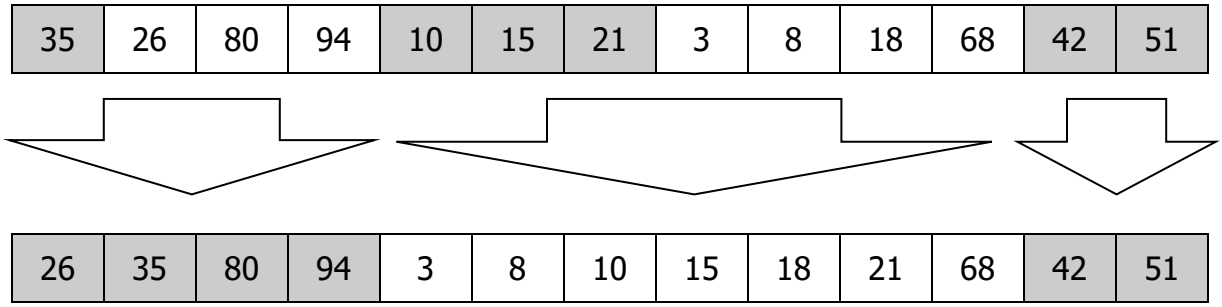
Las **subidas** son los **subarchivos ordenados** y las **bajadas** son el final de una subida y el inicio de otra (cuando la siguiente llave, en vez de ser mayor, es menor).

El número máximo de subidas que puede haber es **n**, y de bajadas, **n-1**. Este caso se presenta si el archivo está ordenado a la inversa.

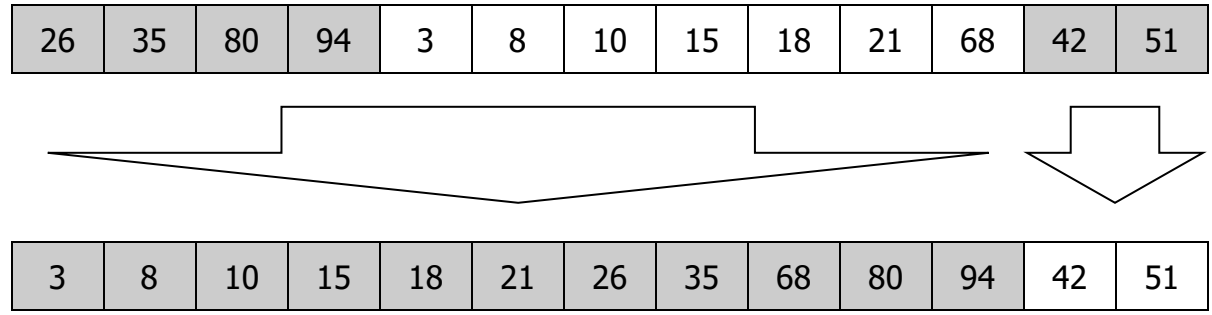
Observe que si un archivo ya se encuentra ordenado, solo habrá una subida y ninguna bajada.

A continuación efectuamos el proceso de ordenamiento de Mezcla Natural en forma completa.

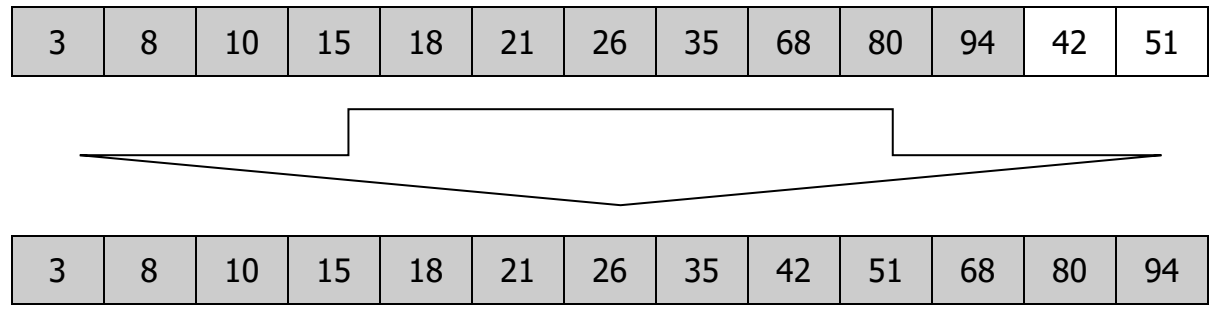
Primera Etapa:



Segunda Etapa:



Tercera (y última) Etapa:



Como podemos ver, se requieren menos etapas con la Mezcla Natural que con la Mezcla Directa, excepto si el archivo estuviera ordenado a la inversa.

El orden de complejidad temporal para este método, también, es **O(n log n)** ya que como sabemos, la notación asintótica se refiere al comportamiento cuando **n** tiende a **infinito**, por esta razón, el orden es el mismo que para Mezcla Directa.

El orden de complejidad espacial es **O(n)**.

7.2 Métodos de Ordenamiento por Distribución (por Paquetes)

Este tipo de métodos se diseñaron para casos en que los datos NO están en un medio de acceso directo (disco duro o memoria de estado sólido).

En el pasado era muy frecuente que los archivos muy grandes se encontraban en cinta magnética (medio de acceso secuencial).

ANALOGÍA
Una persona tiene que ordenar alfabéticamente (por apellidos) un paquete de solicitudes de ingreso de estudiantes a una escuela, el número de solicitudes es grande. Una vez que el encargado de organizar las hojas, las recibe, puede decidir entre: 1. Realizar el ordenamiento considerando a todo el paquete en su conjunto. 2. Tomar una a una las hojas de solicitud y acomodarlas para formar **26 paquetes** de solicitudes independientes (uno para cada letra del alfabeto). Al terminar este procedimiento, los ordenará en forma independiente, finalmente solo restará unir los paquetes ya ordenados.

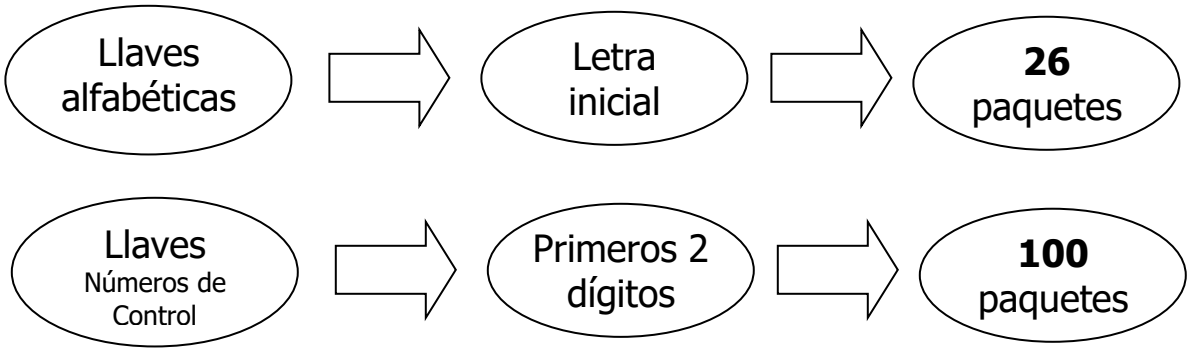
La segunda opción es un método llamado ***ordenamiento por distribución***.

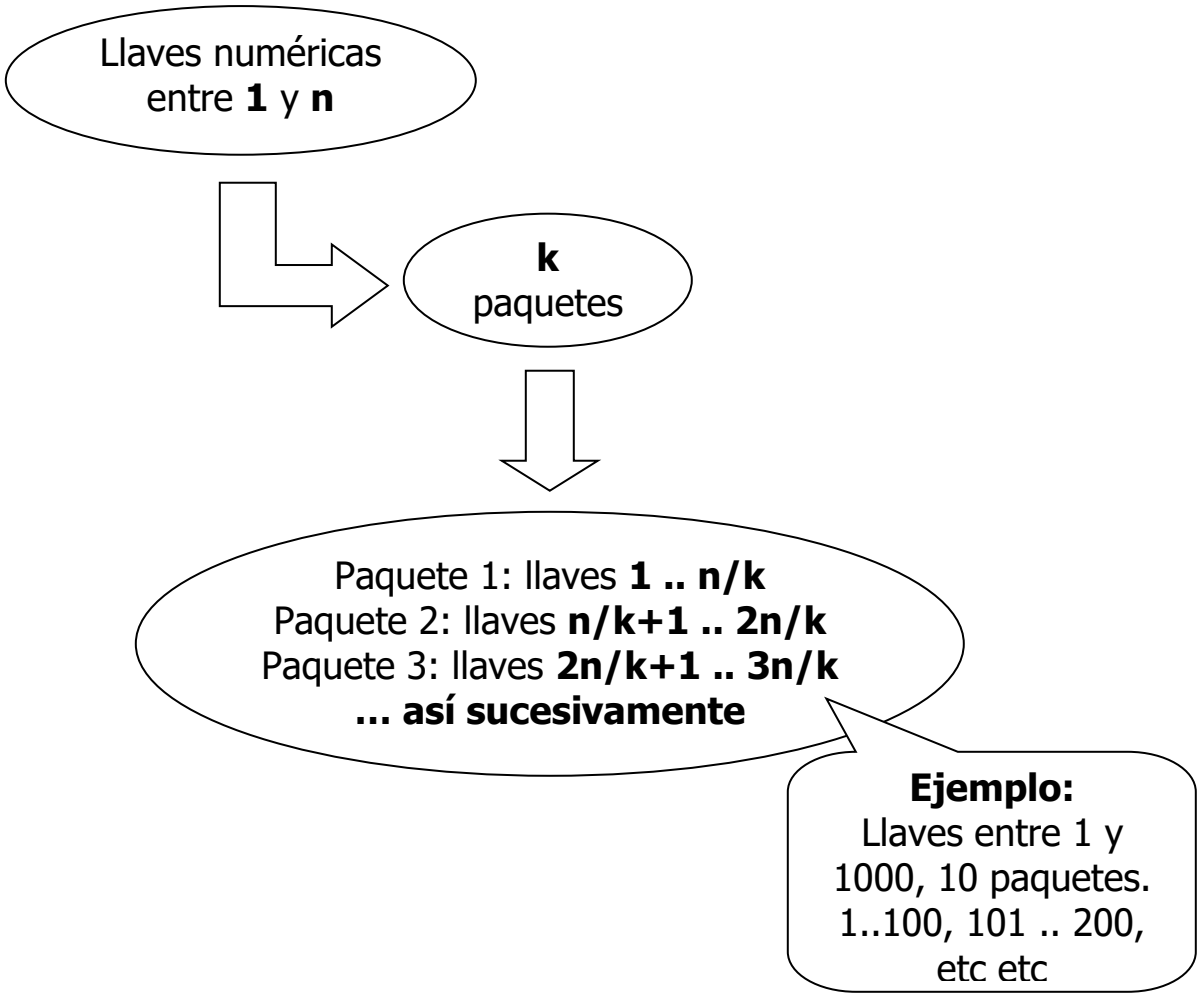
Este tipo de métodos se componen de tres fases:

Distribución en paquetes.
Ordenamiento de los paquetes.
Combinación de los paquetes.

La combinación debe ser un proceso simple (**unión**).

Hay que conocer bien las llaves para elegir como hacer la distribución:



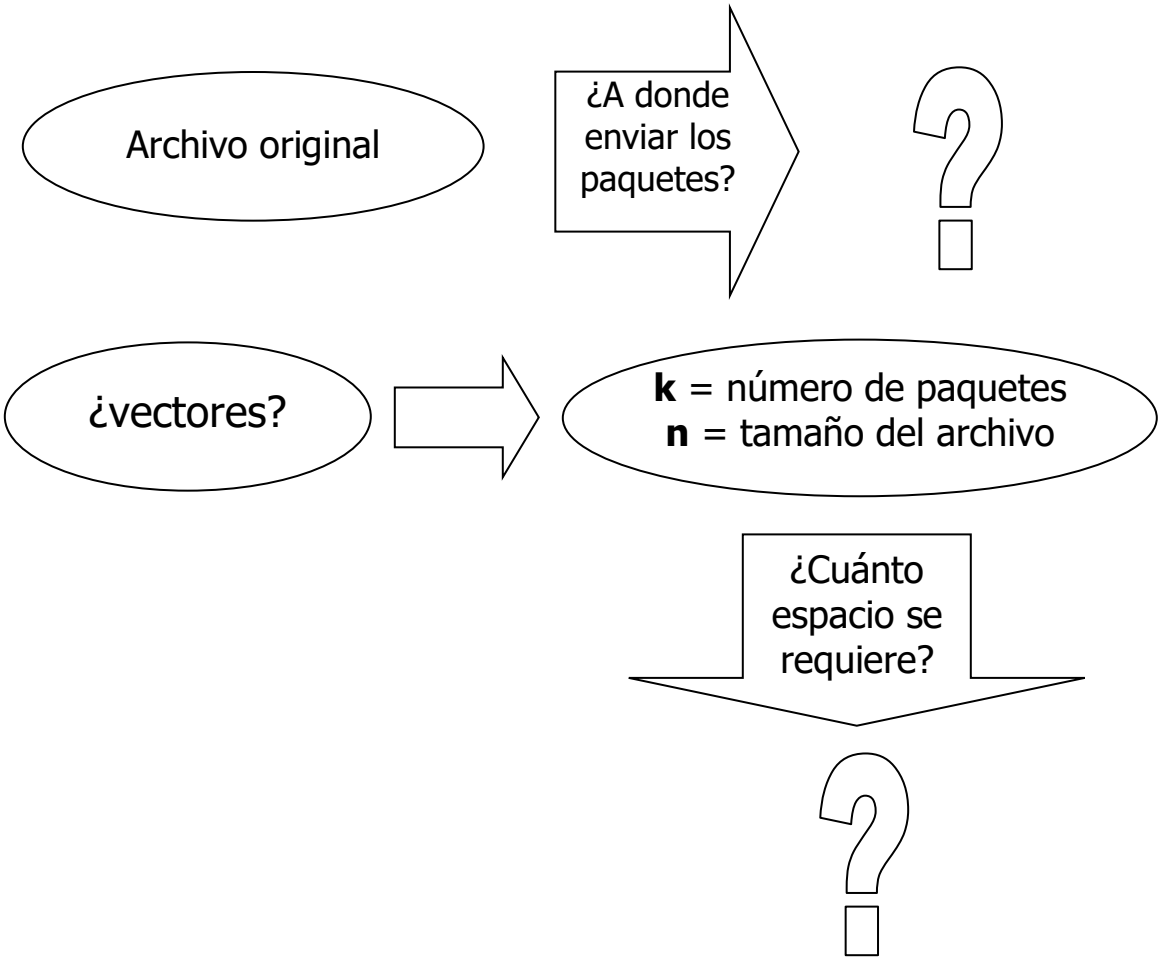

Llaves numéricas
entre **1** y **n**
k
paquetes
Paquete 1: llaves **1 .. n/k**
Paquete 2: llaves **n/k+1 .. 2n/k**
Paquete 3: llaves **2n/k+1 .. 3n/k**
... así sucesivamente
Ejemplo:
Llaves entre 1 y
1000, 10 paquetes.
1..100, 101 .. 200,
etc etc

7.2.1 DISTRIBUCION SIMPLE

Etapas de los métodos de Distribución: **Distribución**, **Ordenamiento**, **Combinación**

Etapa 1. Distribución

Requiere espacio extra. Orden de complejidad espacial **O(n)**



¿Hay una mejor alternativa?

Si.
Listas Encadenadas o **archivos en disco**, por ejemplo

Etapa 2. Ordenamiento

Si se usan listas encadenadas, se puede ordenar el paquete mientras se va formando, evitando la manipulación doble de nodos…

Por lo tanto se fusionan las 2 primeras etapas.

Etapa 3. Combinación

Se toman los registros de cada paquete, uno a uno y se escriben en un archivo con el mismo nombre que el original o en un nuevo archivo. Una vez terminado este proceso, el archivo resultante estará ordenado.

Análisis

¿Por qué distribuir en paquetes?

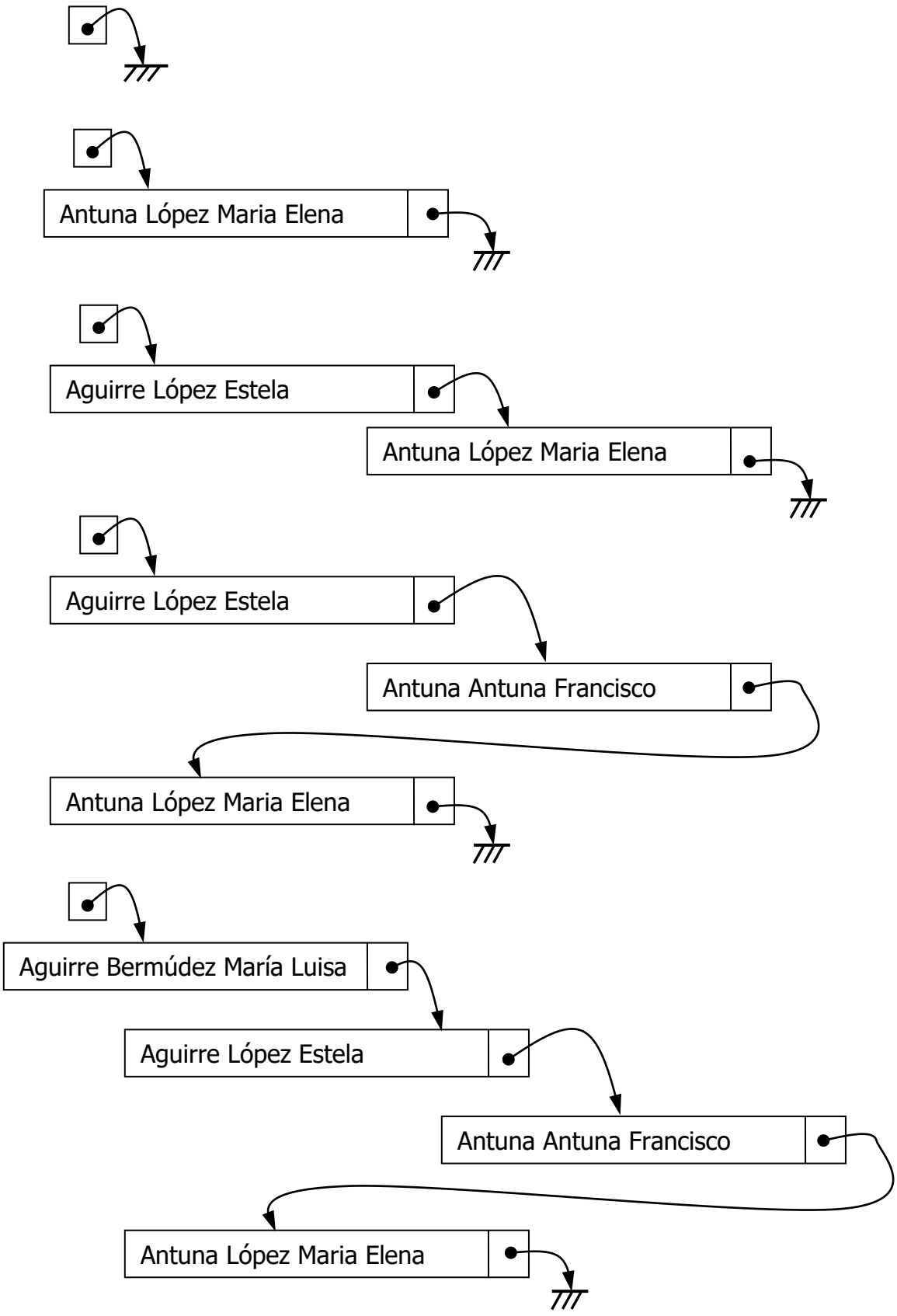
¿Por qué no usar solo una lista encadenada?

- Los archivos pequeños se ordenan con menos esfuerzo que los grandes.
  - El orden de complejidad temporal es **$O(n^2)$** para una lista lineal ordenada.
- 1 archivo de **1000** registros digamos **1'000,000** instrucciones ejecutadas
- 1 archivo de 1000 registros dividido en 26 paquetes:
  - número de registros x paquete ≈ **40**
  - instrucciones ejecutadas x paquete ≈ **1,600**
  - instrucciones acumuladas = 1600 x 26 ≈ **40,000**

Distribución simple ocasiona disminución de trabajo,
aunque el orden de complejidad temporal sigue siendo **$O(n^2)$**

Se llama Distribución Simple porque solo hay **una distribución …**
luego el ordenamiento de los paquetes ….
y finalmente **una combinación**.

Sigamos la formación de la lista de las **"A"** usando una lista encadenada:



Ejercicio 1
● Escribir un programa que cree un archivo que contenga llaves alfabéticas al azar (apellidos y nombre de personas). ● Los apellidos y/o nombres de las personas pueden contener cualquier carácter de la tabla ASCII pero deben iniciar con una letra del alfabeto inglés, es decir, no podrán iniciar con **eñe**, a menos que usted considere esa posibilidad al manipular los paquetes (ejercicio 3).

Ejercicio 2
● Tomar una a una las llaves del archivo del Ejercicio 1 y usando **una** lista encadenada sencilla, ordenarlas y guardarlas ya ordenadas en un archivo con el mismo nombre. ● Habrá que renombrar el archivo original mientras se va creando el nuevo ya ordenado, para eliminar el riesgo de pérdida de datos si se interrumpe el proceso de grabación.

Ejercicio 3
● Usando un **vector** de **listas encadenadas**, realizar el ordenamiento del archivo usando Distribución Simple. ● Se usarán, 26 listas encadenadas, una para cada paquete, de acuerdo a la letra inicial del apellido. ● Al final de la distribución y ordenamiento, el contenido de las listas debe ser vaciado a un archivo con el mismo nombre que el original (este se renombrará). ● Los índices del arreglo, deben ser tipo **Char** (0..25 para las listas 'A' .. 'Z').

7.2.2 Ordenamiento RADIX

Tomemos un archivo con llaves que son **cadenas de 5 dígitos**.

¿Se obtendrían buenos resultados con el siguiente método?

- Distribuir en 10 paquetes de acuerdo al dígito **más** significativo (como si fuera distribución simple).
- No ordenar cada paquete
  - En vez de ello, mientras se hace la primera distribución, hacer otra en base al 2º dígito,
  - y mientras se hacen las 2 primeras hacer una más en base al 3er dígito
  - y así hasta el último dígito.

Se requiere un algoritmo recursivo …

no hay problema … pero … ¿y el espacio requerido?

Los primeros 10 paquetes no pueden ser formados hasta en tanto no se formen los siguientes 100 …

… y estos hasta que no se formen los siguientes 1,000

… Etc

ninguna combinación podrá efectuarse hasta en tanto se realice la última distribución: **(100,000 paquetes)**

¿cómo solucionar este problema?

Empezando por el **dígito menos significativo**

… los paquetes podrán ser combinados
después de cada Distribución

…. las distribuciones se harán de derecha a izquierda

… la necesidad de **comparar** las llaves desaparece

Solo hay **una condición ….**

El orden obtenido en cada etapa **se debe conservar** para el inicio de la siguiente (como en una **COLA**)**.**

Análisis:

De igual forma que Distribución Simple, el número de paquetes depende de la naturaleza de la llave …

Si hay memoria RAM suficiente, conviene usar **Listas Encadenadas** con un **apuntador al final** de la Lista, para que se conserve el orden (como una cola).

Además con las L.E. podremos cambiar el orden de los datos **sin moverlos** en realidad.

Implementación:

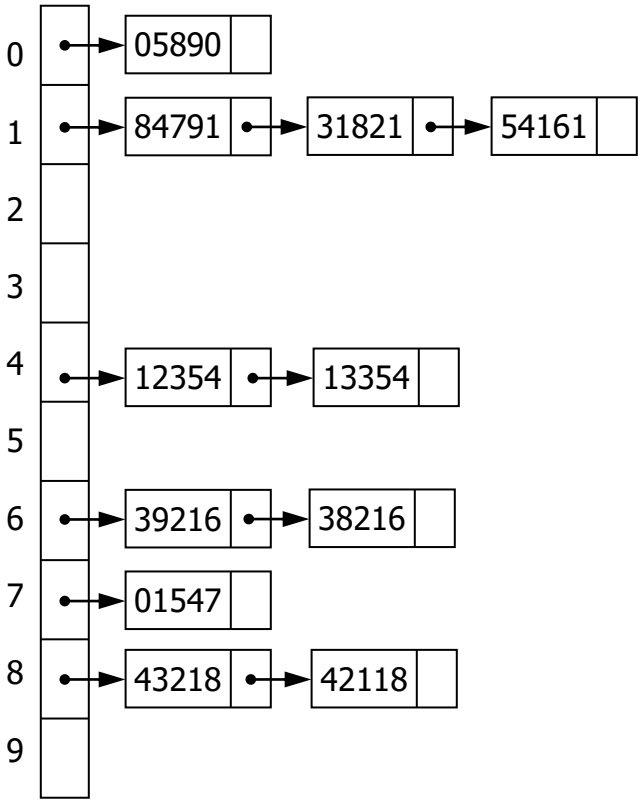
Se requiere una Lista Encadenada para las Combinaciones además de las 10 listas para cada uno de los paquetes.

Ejemplo:

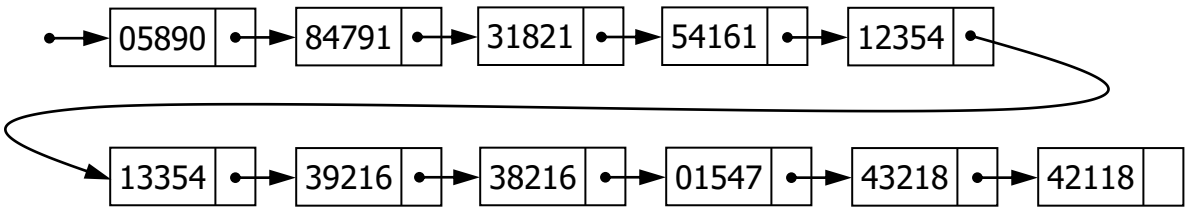
39216	43218	12354	84791	31821	42118	38216	13354	54161	01547	05890

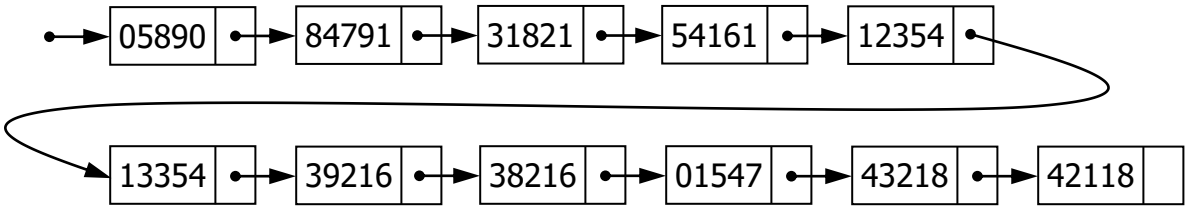
Primera Distribución (Dígito de la Posición 5):

(no se muestra el arreglo de apuntadores al último nodo para simplificar)

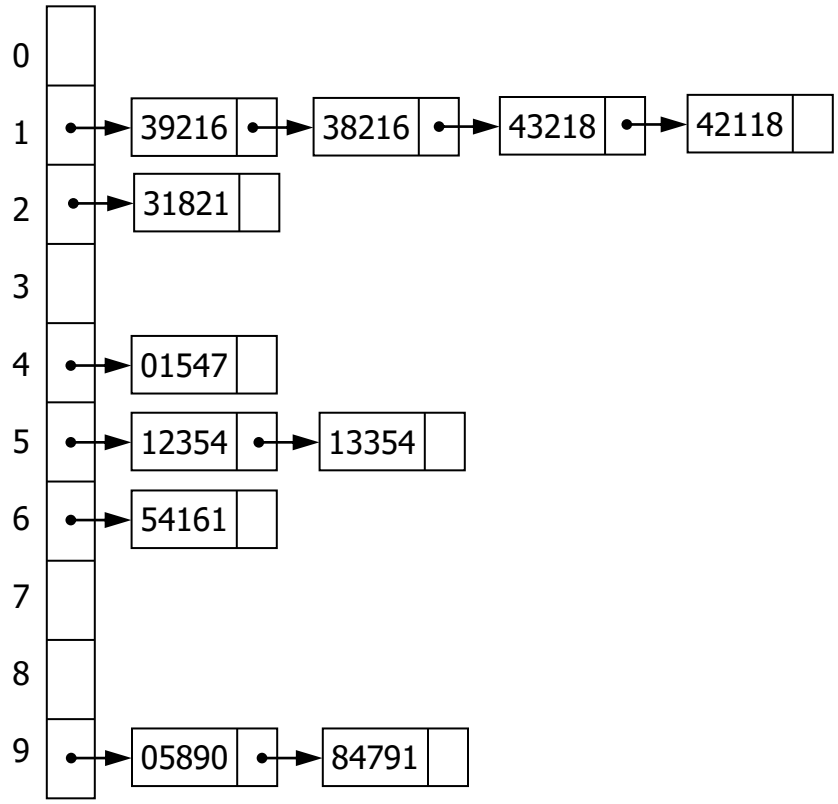


1ª Combinación

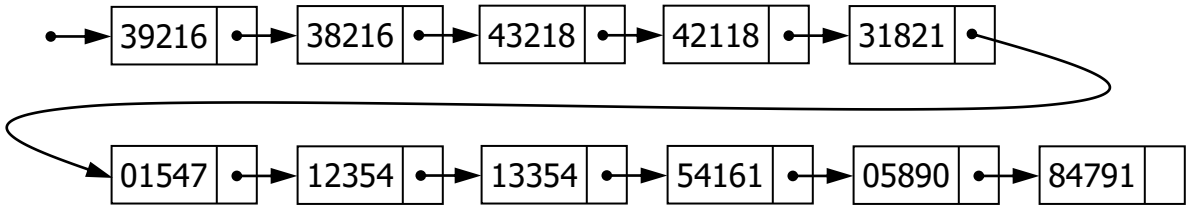


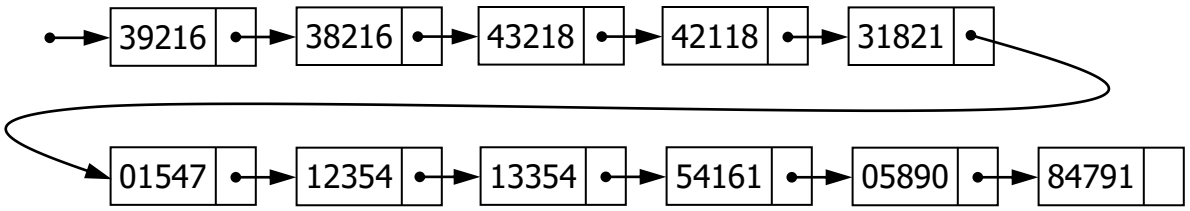


2ª Distribución (4º Dígito):

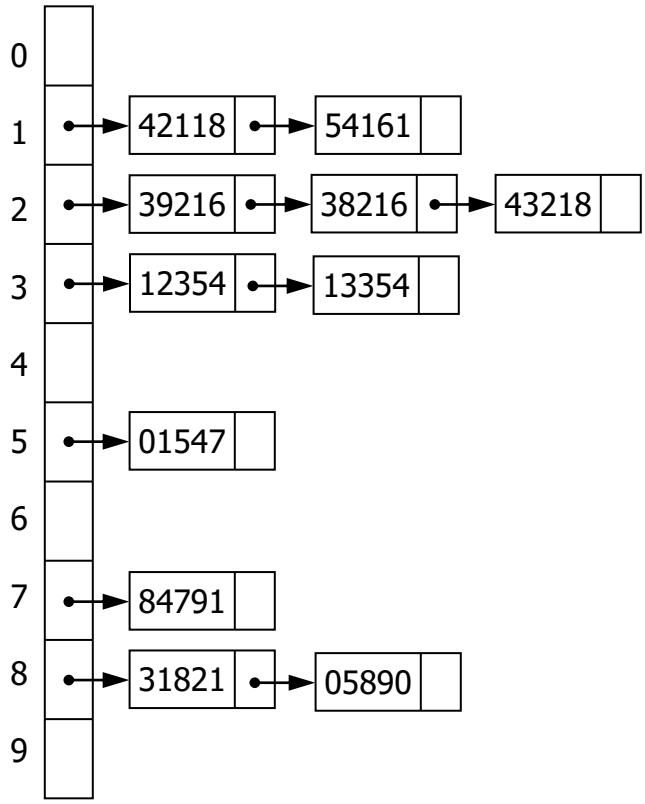


2ª Combinación

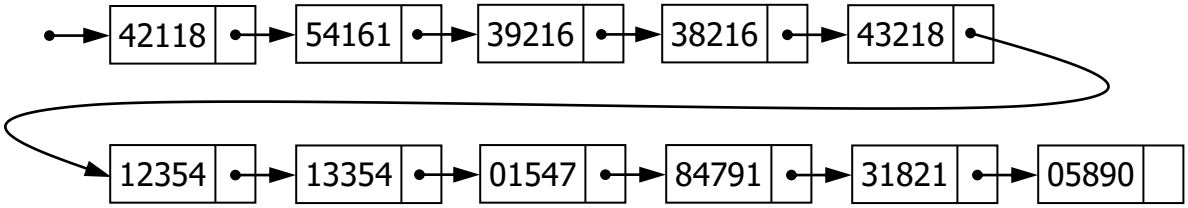


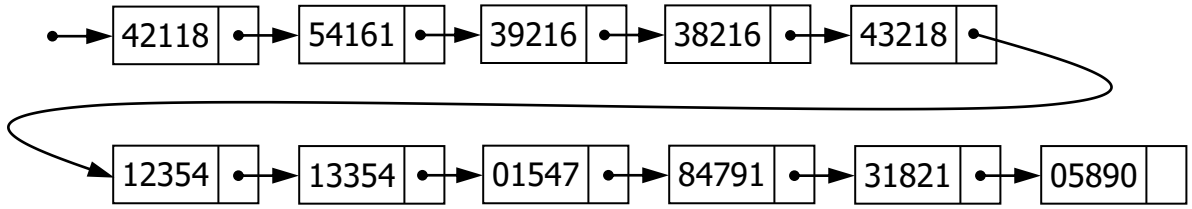


3ª Distribución (3er Dígito):

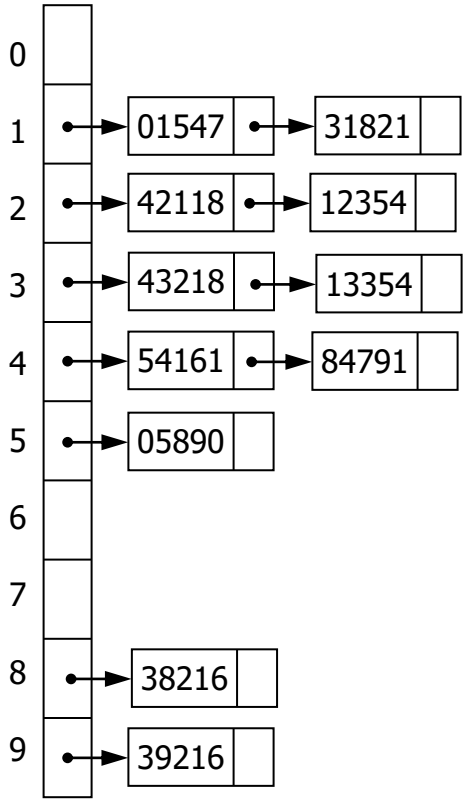


3ª Combinación

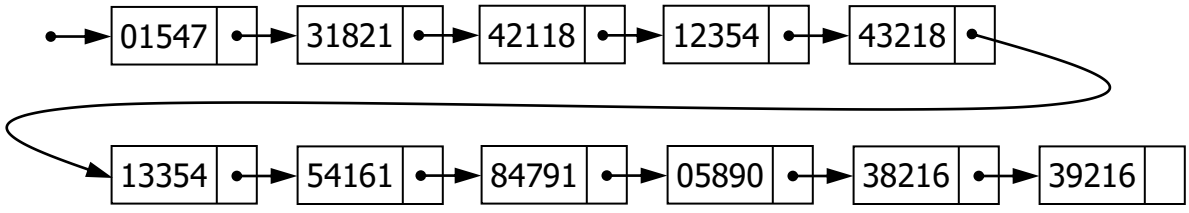


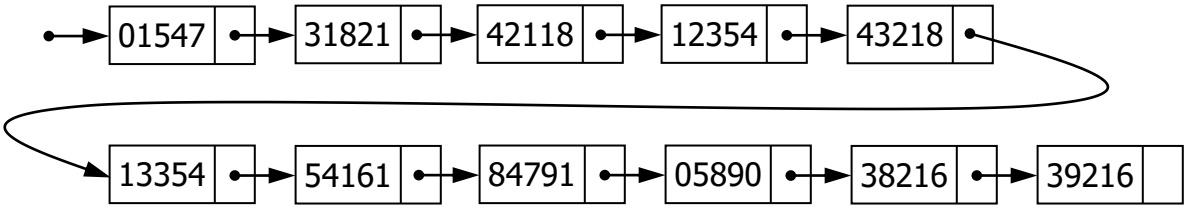


4ª Distribución (2º Dígito):

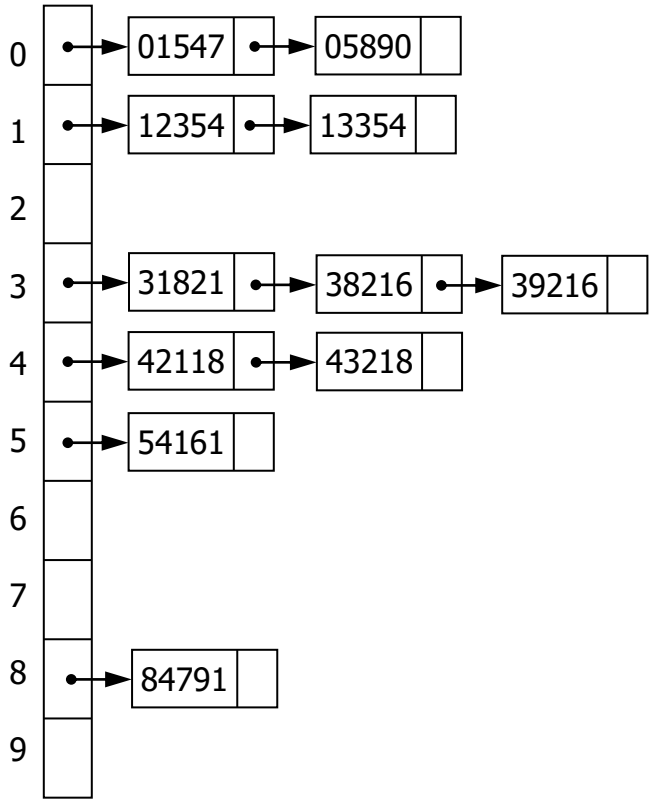


4ª Combinación

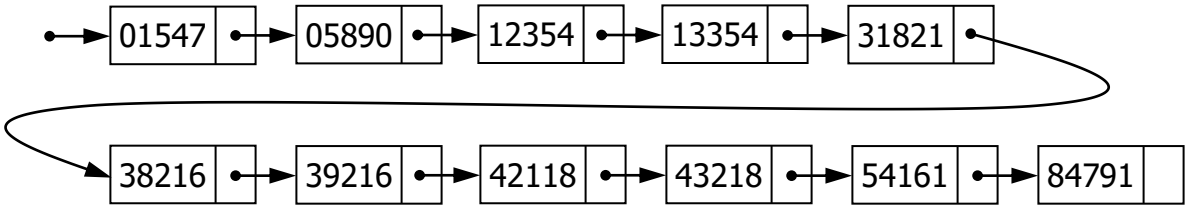




5ª Distribución (1er Dígito):



5ª Combinación



De aquí se pueden escribir los registros al archivo definitivo

ACTIVIDADES DE APRENDIZAJE PARA ESCRIBIR EL PROGRAMA COMPLETO

1. Escribir un programa que Genere **n** llaves ***string*** (combinaciones de dígitos del **0 al 9**, cada llave de **6** caracteres de longitud) al azar y las escriba en un archivo en disco llamado **llaves.dat**.

2. Escribir un programa que tome todas las llaves del archivo **llaves.dat** y las guarde en la memoria en una lista encadenada simple. El orden que las llaves traen del archivo se debe conservar, por lo que la lista encadenada debe contar con un apuntador al último nodo para evitar hacer recorridos cada vez que se añade una nueva llave a la lista encadenada. Una vez terminada la carga de las llaves a la lista, hay que imprimir en la pantalla su contenido.

3. **Hacer una distribución en base al último dígito.**
   Tomando como base el programa anterior, escribir uno nuevo que cuente con **10** listas encadenadas (conviene crear dos arreglos de listas con índices del **0** al **9**). Cada llave que se lee del archivo deberá colocarse en una de las 10 listas dependiendo del último carácter de cada llave, es decir, si la llave termina con el dígito **7** se enlazará a la lista de los **sietes**, y así sucesivamente. Finalmente, imprimir el contenido de las 10 listas.

4. Modificar el programa anterior para que cuente con una lista nueva a la que llamaremos **lista de combinaciones**, su objetivo es **"conectar"** las listas de los paquetes una vez que se hayan leído todas las llaves del archivo. Al final imprima el contenido de la lista de combinaciones para asegurarse que todo funcione correctamente.
   Observe que el paso de los paquetes a la lista de combinaciones solo implica el cambio de 10 apuntadores, independientemente del número de nodos que haya en las listas.

5. Complete **RADIX** tomando como base el programa anterior (punto 4) para que se repitan las distribuciones y las combinaciones tantas veces como dígitos tenga la llave (considerando cada dígito según corresponda).

Observaciones

- No conviene crear y destruir nodos continuamente mientras se transfieren de una lista a otra, solo hay que cambiar los apuntadores. Hacerlo así redunda en un ahorro muy importante de instrucciones ejecutadas.
- Si la llave está formada por caracteres cualquiera, es importante asegurarse de que todas las llaves sean de la misma longitud.
- El razonamiento, para este método, en cuanto a los espacios a la derecha en llaves tipo cadena es el mismo respecto a los espacios a la izquierda en las llaves numéricas.