**El merge tiene una terminologia especial:**  
Cada archivo de entrada se llama particion. Como podemos ver tiene 'n+1' cantidad de archivos abiertos, y como sabemos un SO no puede abrir una cantidad infinita de archivos. Cada archivo abierto se llama una 'via'. Entonces, hay un maximo de vias, que viene a ser una entrada de registros y una salida de registros, de ahi la palabra 'via'. Un merge para tener logica tiene que tener por lo menos dos vias de entrada y una de salida. Cuando la cantidad de archivos a intercalar supera la cantidad de vias posibles, entonces no podemos hacerlo en una sola etapa. Tenemos que hacer un merge primero y despues volverlo a intentar con los archivos que quedaron. Cada pasada se llama una 'fase' del merge.

Si quisieramos hacer el merge en una sola fase entonces la cantidad de archivos a intercalar tiene que ser igual a la cantidad de vias como maximo. La cantidad de vias menos uno me da el maximo de particiones que se pueden intercalar. Nos puede pasar que nos quede un solo archivo para intercalar y haya que leer todo el merge previo para poder intercalar.  
Criterio de eficiencia, tenemos que buscar la manera de leer la menor cantidad de veces los registros.  
En esta operación de merge la cuestion de la memoria no es algo critico.  
Como se está buscando un criterio de eficiencia nos aparecen distintas formas de hacer el merge (3). Estas formas tienen sentido cuando hay mas particiones que vías.

**Sort interno:**Se lee todo el archivo dentro de la memoria, se lo ordena y se lo graba. Tema que ya vieron en otra materia, en este caso se usa uno de los metodos conocidos para ordenar.  
El problema empieza cuando el archivo no entra en memoria.

*Una primera solucion seria el 'Key sort'.* Lo que decimos es, en el fondo no tenemos por que meter todo el registro completo en memoria, porque lo unico que nos importa para ordenar son las claves. Pero cuando ordenemos tenemos que poder juntar la clave con el resto del registro y para eso necesitamos poder hacer acceso directo.  
  
(este metodo funciona si las claves caben en memoria)  
-Pasamos las claves a memoria  
  
-Las ordenamos  
  
-Accedemos a ese registro con cada una de esas claves.  
Tiene que estar en un archivo relativo o en un stream.  
  
Una vez que ordenamos las claves accedemos por clave al registro y lo mandamos al archivo de salida. Ese es el costo del sistema. La primer lectura es secuencial, no representa un gran costo en acceso a disco. Pero la segunda si, porque por cada clave tenemos que hacer un seek.  
  
Cual es la limitacion? Pueden existir casos en que las claves no quepan en memoria, entonces estamos de vuelta en el caso anterior.

Surgen otros métodos Replacement selection y natural selection.

**Remplacement selection**

**(DEBIDO A Knuth com de ACM, 1965)**

            Sea m el número de registros que se van a ordenar que se pueden mantener en la memoria principal. Imagina que estas posiciones de memoria son registros y asumen podemos marcarlos como "on" u "off". Selección del reemplazo puede superponerse a leer, ordenar y escribir.

**El Algoritmo:**

***Paso 1****:* Los m registros están llenos de registros de la entrada que realizar la clasificación *.*

***Paso 2:*** Todos los registros se colocan en el estado "encendido".

***Paso 3:*** Seleccione el registro que tiene el más pequeño de todos "en" registros.

***Paso 4:*** Verter el contenido del registro seleccionado a la salida (llámese como tecla Y).

***Paso 5:*** Vuelva a colocar el contenido del registro seleccionado por el siguiente registro de entrada.

            Si la llave de registro> Y

                        Vaya al paso 3

            Si la llave de registro == Y

                        Vaya al paso 4

            Si la llave de registro <Y

                        Vaya al paso 6

***Paso 6:*** Encienda el registro de claves seleccionado   "off".

            Si ahora todos los registros son   "off"

                        Hemos completado una subcadena ordenada (pista).

                        Comenzamos un nuevo grupo subcadena e ir al paso 2.

            Más

                        Vaya al paso 3

**Ejemplo:** Supongamos que el número de registros es 3 y teclas de entrada son 5, 2, 9, 7, 0, 8, 1, 6, 3, 4 ...

**Los registros**                               **de salida**

            5 2 9                                        2

            5 7 9                                        5

\* 0 7 9                                      7

\* 0 8 9                                      8

\* 0 \* 1 9                                     9

\* 0 \* 1 \* 6

0 1 6                                        0

3 1 6                                        1

3 4 6                                        3

- 4 6                                         4

- 6                                         6

---

Natural Selection:  
Tenemos el mismo archivo:  
  
2, 1, 3, 5, 7, 8, 2, 6, 4, 1, 2, 0, 3, 5  
En este metodo cuando leemos una clave que es menor que la que ya salio, en lugar de dejarla en memoria la mandamos a un archivo a disco (lo llamamos freezer) tiene que ser del mismo tamaño de la memoria, cuando el freezer esta lleno creamos una particion llena. Mas o menos como antes pero no ocupamos el tamaño ese espacio en memoria cuando hay que congelar uno. Pero de todas formas el tamaño del freezer es igual al de la memoria.  
  
Ganamos un slot mas pero tenemos el problema de tener que mandar a disco y leer de disco.  
Memoria  
  
|\_|\_|\_|  
Freezer (en disco)  
  
|\_|\_|\_|  
Leemos los 3 primeros  
  
2,1,3  
  
Sale 1, entra 5, sale 2, entra 7, sale 3, entra 8, sale 5, entra 2 (pero el 2 es menor! asi que va al freezer) leo el siguiente, el 6, leo el 4 (es menor! Al freezer carajo!), leo el siguiente el 1 (Menor! Al freezer mierda!)  
  
Se lleno el freezer, cagamos. mandamos los que quedaron. La particion queda:  
  
1,2,3,5,6,7,8  
  
La particion quedo mas grande que antes, pudimos meter el 6.  
Nueva particion:  
  
Vaciamos el freezer y mandamos todo a memoria.  
  
2,4,1  
  
Volvemos a empezar.  
  
Sale 1, leemos 2, sale el 2 leemos el 0 (al freezer), leemos el 3, sale el 2 entra el 5, sale el 3, sale el 4, sale el 5.  
  
Tengo la segunda particion:  
  
1,2,2,3,4,5  
Me quedo un cero  
  
Nueva particion:  
  
Mando el cero del freezer a la memoria, y lo mando a la particion. Fin.  
  
Tercera particion:  
  
0  
En este sistema en promedio tomando en cuenta que las claves esten distribuidas al azar, el tamaño de la particion es 3 veces la memoria (en promedio, a lo mejor no hay ninguna que sea 3 veces la memoria)  
La desventaja de este metodo es que tiene que trabajar con el disco. Nuevamente valen los razonamientos anteriores, si el archivo ya esta ordenado no se usa el freezer y salen como estan, si el archivo esta ordenado a la inversa cagaste. :P