

El procedimiento inicial

MemoriaSecundariaMergesort(*I*)

$r := 0$

$k := \lfloor \frac{M}{B} \rfloor$

$\acute{u}ltimo := 0$

$pos := 1$

LeerArchivo(*I*, *pos*, *k*, *L*)

while Length(*L*) > 0 **do**

Quicksort(*L*, 1, Length(*L*))

EscribirArchivo(*O*, *pos*, *L*)

$r := r + 1$

$pos := pos + k$

$\acute{u}ltimo := \lceil \frac{\text{Length}(L)}{B} \rceil$

LeerArchivo(*I*, *pos*, *k*, *L*)

intercambiar *I* con *O*

return MemoriaSecundariaMergesort(*I*, *O*, *r*, *k*, $\acute{u}ltimo$)

Nota: en este procedimiento *I* y *O* son punteros a archivos, y suponemos que el archivo apuntado por *I* contiene al menos un número entero

El procedimiento inicial: un ejemplo

Supongamos que $N = 66$, $M = 13$ y $B = 4$

► Tenemos que $k = \lfloor \frac{M}{B} \rfloor = 3$

El archivo de entrada I es de la forma:

$$\begin{array}{c} m_1 \\ m_2 \\ \vdots \\ m_{66} \end{array}$$

El procedimiento inicial: un ejemplo

Inicialmente $pos = 1$, por lo que la llamada **LeerArchivo**(l, pos, k, L) deja en L la lista $[m_1, m_2, \dots, m_{16}]$

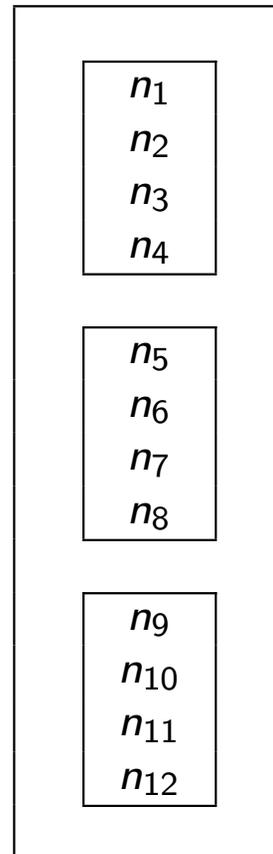
- ▶ Esta llamada lee tres bloques con cuatro enteros cada uno

Utilizamos **Quicksort** para ordenar L de menor a mayor.

- ▶ Supongamos que el resultado es $[n_1, n_2, \dots, n_{16}]$, el cual es almacenado en la misma lista L

El procedimiento inicial: un ejemplo

La llamada **EscribirArchivo**(O , pos , L) escribe en el archivo O lo siguiente:



En esta figura usamos los rectángulos interiores para mostrar los bloques.

El procedimiento inicial: un ejemplo

Al salir del loop tenemos que $r = 6$ y $\text{último} = 2$

- ▶ Decimos entonces que tenemos 6 **runs**
 - ▶ Los primeros 5 runs tienen 3 bloques, el último run tiene 2 bloques
- ▶ El archivo O tiene $(r - 1) \cdot k + \text{último} = 5 \cdot 3 + 2 = 17$ bloques

Finalmente intercambiamos I y O , y realizamos la llamada **MemoriaSecundariaMergesort**($I, O, 6, 3, 2$)

- ▶ I sigue siendo el archivo de entrada y O el archivo de salida

Vamos a definir el procedimiento **MemoriaSecundariaMergesort** que recibe 5 parámetros como entrada

El procedimiento principal

```
MemoriaSecundariaMergesort( $I, O, r, k, \text{último}$ )
  if  $r = 1$  then return  $I$ 
  else
     $p_1 := 1$ 
    while  $p_1 \leq (r - 3) \cdot k + 1$  do
       $p_2 := p_1 + k$ 
      Merge( $I, O, p_1, k, p_2, k$ )
       $p_1 := p_1 + 2 \cdot k$ 
     $p_2 := p_1 + k$ 
    if  $p_2 \leq (r - 1) \cdot k + \text{último}$  then
      Merge( $I, O, p_1, k, p_2, \text{último}$ )
       $\text{último} = k + \text{último}$ 
    else
      LeerArchivo( $I, p_1, \text{último}, L$ )
      EscribirArchivo( $O, p_1, L$ )
    intercambiar  $I$  con  $O$ 
  return MemoriaSecundariaMergesort( $I, O, \lceil \frac{r}{2} \rceil, 2 \cdot k, \text{último}$ )
```

El procedimiento principal: continuación del ejemplo

Tenemos que $(r - 3) \cdot k + 1 = 3 \cdot 3 + 1 = 10$

Por lo tanto el ciclo **while** realiza las siguientes llamadas:

- ▶ **Merge**(I , O , 1, 3, 4, 3): mezcla los bloques 1, 2, 3 con los bloques 4, 5, 6 del archivo I para dejar 6 bloques de números enteros ordenados (de menor a mayor) en O desde la posición 1
- ▶ **Merge**(I , O , 7, 3, 10, 3): mezcla los bloques 7, 8, 9 con los bloques 10, 11, 12 del archivo I para dejar 6 bloques de números enteros ordenados en O desde la posición 7
 - ▶ Vale decir, los 6 bloques son puestos en O a continuación de los 6 bloques ordenados generados por **Merge**(I , O , 1, 3, 4, 3)

El procedimiento principal: continuación del ejemplo

Tenemos que $(r - 1) \cdot k + \text{último} = 5 \cdot 3 + 2 = 17$, por lo que la llamada **Merge**(I , O , 13, 3, 16, 2) es efectuada después del ciclo **while**

- ▶ Esta llamada mezcla los bloques 13, 14, 15 con los bloques 16, 17 del archivo I para dejar 5 bloques de números enteros ordenados en O desde la posición 13

Finalmente tenemos que $\lceil \frac{r}{2} \rceil = 3$, $2 \cdot k = 2 \cdot 3 = 6$ y $\text{último} = 3 + 2 = 5$

- ▶ Por lo tanto tenemos 3 runs en O , los dos primeros con 6 bloques y el último con 5 bloques

Los punteros I y O son intercambiados y se realiza la llamada **MemoriaSecundariaMergesort**(I , O , 3, 6, 5)

- ▶ I es nuevamente el archivo de entrada, pero ahora contiene 3 runs ordenados

El procedimiento principal: continuación del ejemplo

Tenemos ahora que $(r - 3) \cdot k + 1 = 0 \cdot 6 + 1 = 1$

Por lo tanto el ciclo **while** sólo realiza la siguientes llamada:

- ▶ **Merge**(I , O , 1, 6, 7, 6): mezcla los bloques 1, 2, 3, 4, 5, 6 con los bloques 7, 8, 9, 10, 11, 12 del archivo I para dejar 12 bloques de números enteros ordenados en O desde la posición 1

El procedimiento principal: continuación del ejemplo

Después de salir del ciclo **while** tenemos que $p_1 = 13$ y $p_2 = 19$, por lo que $p_2 > (r - 1) \cdot k + \text{último} = 2 \cdot 6 + 5 = 17$ y las siguientes llamadas son realizadas:

- ▶ **LeerArchivo**($I, 13, 5, L$): deja en la lista L los enteros en los últimos 5 bloques de I , los cuales están ordenados de menor a mayor
- ▶ **EscribirArchivo**($O, 13, L$): deja los 5 bloques almacenados en L en el archivo O desde la posición 13
 - ▶ Vale decir, deja los 5 bloques a continuación de los 12 bloques generados por **Merge**($I, O, 1, 6, 7, 6$)

El procedimiento principal: continuación del ejemplo

Finalmente tenemos que $\lceil \frac{r}{2} \rceil = 2$, $2 \cdot k = 2 \cdot 6 = 12$ y *último* = 5

- ▶ Por lo tanto tenemos 2 runs en *O*, el primero con 12 bloques y el segundo con 5 bloques

Los punteros *I* y *O* son intercambiados y se realiza la llamada **MemoriaSecundariaMergesort**(*I*, *O*, 2, 12, 5)

- ▶ *I* es nuevamente el archivo de entrada, pero ahora contiene 2 runs ordenados

El procedimiento principal: continuación del ejemplo

Tenemos ahora que $(r - 3) \cdot k + 1 = -1 \cdot 12 + 1 = -11$

- ▶ Por lo tanto no entramos al ciclo **while** y tenemos que $p_1 = 1$ y $p_2 = 13$

Dado que $p_2 \leq (r - 1) \cdot k + \text{último} = 1 \cdot 12 + 2 = 17$ se realiza la siguiente llamada:

- ▶ **Merge**($I, O, 1, 12, 13, 5$): mezcla los bloques 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 con los bloques 13, 14, 15, 16, 17 del archivo I para dejar 17 bloques de números enteros ordenados en O desde la posición 1

El procedimiento principal: continuación del ejemplo

Finalmente tenemos que $\lceil \frac{r}{2} \rceil = 1$, $2 \cdot k = 2 \cdot 12 = 24$ y $\text{último} = 12 + 5 = 17$

- ▶ Por lo tanto tenemos 1 run en O con 17 bloques

Los punteros I y O son intercambiados y se realiza la llamada **MemoriaSecundariaMergesort**(I , O , 1, 24, 17)

- ▶ I es nuevamente el archivo de entrada, pero ahora contiene 1 run ordenado

Como $r = 1$ en la llamada **MemoriaSecundariaMergesort**(I , O , 1, 24, 17), el puntero I es retornado

- ▶ I es el resultado de las llamadas a **MemoriaSecundariaMergesort**, en particular es retornado por **MemoriaSecundariaMergesort**(I , O , 6, 3, 2) y **MemoriaSecundariaMergesort**(I)

El procedimiento para mezclar: valores iniciales

Merge($I, O, pos_1, r_1, pos_2, r_2$)

$p := pos_1$

$p_1 := pos_1$

$p_2 := pos_2$

$L_1 := \text{TransformarLista}(\text{LeerMemoriaSecundaria}(I, p_1))$

$L_2 := \text{TransformarLista}(\text{LeerMemoriaSecundaria}(I, p_2))$

$L := \emptyset$

$i_1 := 1$

$i_2 := 1$

$j := 1$

Mezclando desde ambos runs

```
while  $p_1 \leq pos_1 + r_1 - 1$  and  $p_2 \leq pos_2 + r_2 - 1$  do  
  while  $i_1 \leq \text{Length}(L_1)$  and  $i_2 \leq \text{Length}(L_2)$  and  $j \leq B$  do  
    if  $L_1[i_1] \leq L_2[i_2]$  then  
       $L[j] := L_1[i_1]$   
       $i_1 := i_1 + 1$   
    else  
       $L[j] := L_2[i_2]$   
       $i_2 := i_2 + 1$   
     $j := j + 1$   
  if  $j = B + 1$  then  
    EscribirMemoriaSecundaria( $O, p, \text{TransformarBloque}(L)$ )  
     $L := \emptyset$   
     $j := 1$   
     $p := p + 1$ 
```

Mezclando desde ambos runs

```
if  $i_1 > \text{Length}(L_1)$  then
   $p_1 := p_1 + 1$ 
  if  $p_1 \leq pos_1 + r_1 - 1$  then
     $L_1 := \text{TransformarLista}(\text{LeerMemoriaSecundaria}(I, p_1))$ 
     $i_1 := 1$ 
if  $i_2 > \text{Length}(L_2)$  then
   $p_2 := p_2 + 1$ 
  if  $p_2 \leq pos_2 + r_2 - 1$  then
     $L_2 := \text{TransformarLista}(\text{LeerMemoriaSecundaria}(I, p_2))$ 
     $i_2 := 1$ 
```

Mezclando cuando el primer run está terminado

```
if  $p_1 > pos_1 + r_1 - 1$  then
  while  $p_2 \leq pos_2 + r_2 - 1$ 
    while  $i_2 \leq \text{Length}(L_2)$  and  $j \leq B$  do
       $L[j] := L_2[i_2]$ 
       $i_2 := i_2 + 1$ 
       $j := j + 1$ 
    if  $j = B + 1$  then
      EscribirMemoriaSecundaria( $O, p, \text{TransformarBloque}(L)$ )
       $L := \emptyset$ 
       $j := 1$ 
       $p := p + 1$ 
    if  $i_2 > \text{Length}(L_2)$  then
       $p_2 := p_2 + 1$ 
      if  $p_2 \leq pos_2 + r_2 - 1$  then
         $L_2 := \text{TransformarLista}(\text{LeerMemoriaSecundaria}(I, p_2))$ 
         $i_2 := 1$ 
  if  $j > 1$  then
    EscribirMemoriaSecundaria( $O, p,$ 
      TransformarBloque(SubList( $L, 1, j - 1$ )))
```

Mezclando cuando el segundo run está terminado

else

while $p_1 \leq pos_1 + r_1 - 1$

while $i_1 \leq \text{Length}(L_1)$ and $j \leq B$ do

$L[j] := L_1[i_1]$

$i_1 := i_1 + 1$

$j := j + 1$

if $j = B + 1$ then

EscribirMemoriaSecundaria($O, p, \text{TransformarBloque}(L)$)

$L := \emptyset$

$j := 1$

$p := p + 1$

if $i_1 > \text{Length}(L_1)$ then

$p_1 := p_1 + 1$

if $p_1 \leq pos_1 + r_1 - 1$ then

$L_1 := \text{TransformarLista}(\text{LeerMemoriaSecundaria}(I, p_1))$

$i_1 := 1$

if $j > 1$ then

EscribirMemoriaSecundaria($O, p,$
TransformarBloque(**SubList**($L, 1, j - 1$)))