

Tema 3
CLASIFICACIÓN
EN
MEMORIA
SECUNDARIA

Tutor: Enrique J. Carmona Suárez
Asignatura: "Estructuras de Datos y Algoritmos"

TEMA III : CLASIFICACIÓN EN MEMORIA SECUNDARIA

3.1. Clasificación externa basada en mezcla

3.1.1. Mezcla directa.

3.1.2. Mezcla natural.

2.1.3. Mezcla balanceada múltiple.

3.1.4. Clasificación polifásica.

3.2. Archivos Indexados

3.3 Tablas de dispersión (Hashing)

CLASIFICACIÓN EN MEMORIA SECUNDARIA

Problema: Los datos a ordenar no caben todos en memoria principal , están almacenados en dispositivos periféricos de acceso secuencial:

- Cinta: TDA secuencia
- Disco: TDA archivo

Restricciones en el proceso de clasificación:

- Acceso secuencial a cada uno de los elementos de una secuencia.
- No permite aplicar los métodos de clasificación sobre arreglos en los que el acceso a cualquier elemento implicaba el mismo coste.

Tipos de técnicas de clasificación externa basadas en Mezcla:

- Mezcla directa
- Mezcla directa de una fase
- Mezcla natural
- Mezcla balanceada múltiple
- Clasificación polifásica

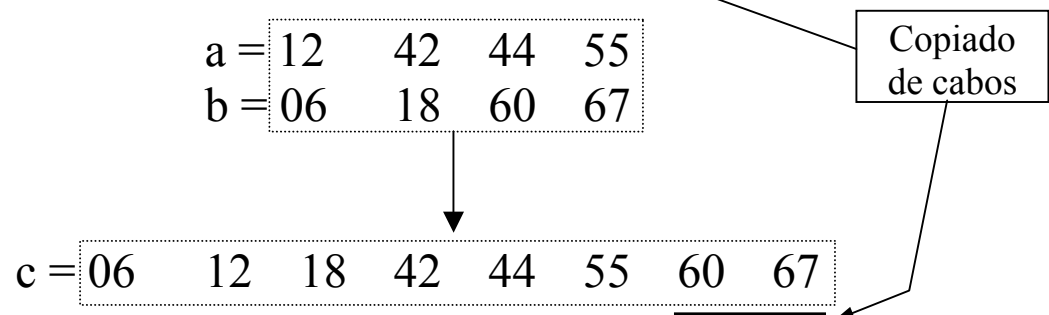
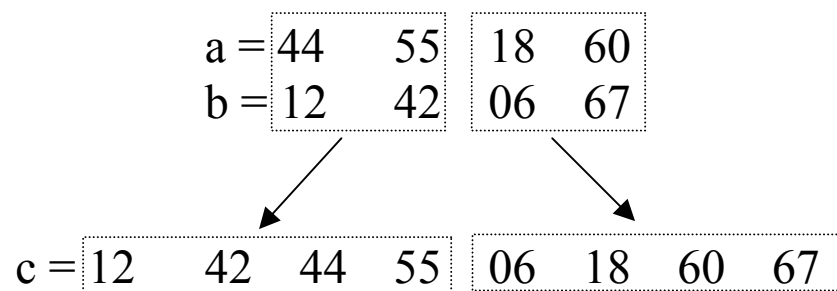
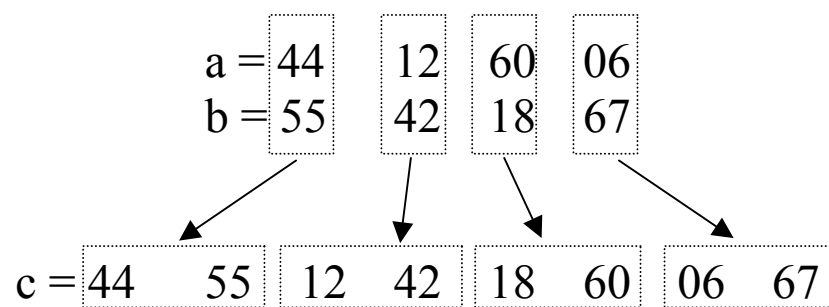
Definiciones:

- **Mezcla:** Tarea de combinar varias secuencias en una sola, mediante la selección repetida de los componentes accesibles en cada momento. Es una operación auxiliar previa utilizada como estrategia para desarrollar la tarea de clasificación.
- **Fase:** Cada operación que trata un conjunto completo de datos. Ejemplos: *distribución, mezcla*.
- **Pase (etapa):** el proceso más corto que, por repetición, forma parte del proceso de clasificación. Ejemplo: *distribución + mezcla*.
- **Cinta:** cada una de las secuencias necesarias en el proceso de clasificación.

Mezcla Directa

1. Dividir la secuencia original, c , en dos mitades, a y b .
2. Mezclar a y b combinando cada elemento en pares ordenados.
3. Llamar c a la secuencia mezclada y repetir los pasos 1 y 2, combinando los pares ordenados en cuádruplos ordenados.
4. Seguir haciendo esto (duplicando las longitudes de las subsecuencias combinadas) hasta que quede ordenada toda la secuencia.

$c = 44 \quad 55 \quad 12 \quad 42 \quad 60 \quad 18 \quad 06 \quad 67$



Fase de división

```
PROCEDURE Separa(VAR c,a,b: File;p: CARDINAL;  
VAR longc: CARDINAL);  
  (*Divide c en a y b en grupos de p elementos (1,2,4,8...) y  
  en longc devuelve la longitud de c*)  
  VAR  
    i,j: CARDINAL;  
  
  BEGIN  
    Reset(c);  
    Create(a,'nuevoa.dat');  
    Create(b,'nuevob.dat');  
    i:=0; j:=0; longc:=0;  
    ReadWord(c,w);  
    WHILE ~c.eof DO  
      WHILE (~c.eof) & (i<p) DO  
        WriteWord(a,w);  
        ReadWord(c,w);  
        i:=i+1;  
        longc:=longc+1;  
      END;  
      i:=0;  
      WHILE (~c.eof)& (j<p) DO  
        WriteWord(b,w);  
        ReadWord(c,w);  
        j:=j+1;  
        longc:=longc+1;  
      END;  
      j:=0;  
    END;  
  END Separa;
```

Fase de Mezcla

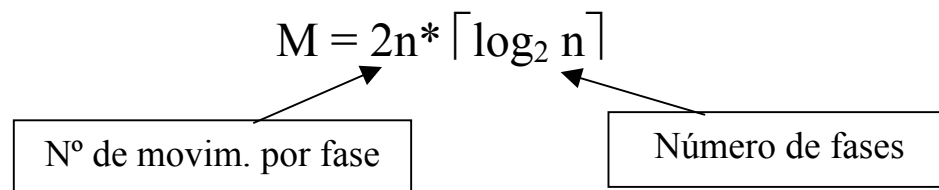
```
PROCEDURE mezcla(VAR a, b, c: File; p, longc: CARDINAL);
(*Mezcla a y b en c en grupos ordenados de 2p elementos*)
VAR q, (*nº elementos que quedan por mezclar de la 1ª subsec*)
    r, (*nº elementos que quedan por mezclar de la 2ª subsec*)
    m: CARDINAL; (*nº de elementos por mezclar*)
    wa, wb: INTEGER;
BEGIN
    Reset(a); Reset(b); Reset(c); m:=longc;
    REPEAT
        IF m>=p THEN q:=p ELSE q:=m END;
        m:=m-q;
        IF m>=p THEN r:=p ELSE r:=m END;
        m:=m-r;
        (* Proposicion de mezcla *)
        ReadWord(a,wa); ReadWord(b,wb);
        WHILE (q#0)&(r#0) DO
            IF wa<wb THEN
                WriteWord(c,wa); q:=q-1;
                IF q#0 THEN ReadWord(a,wa); END
            ELSE
                WriteWord(c,wb); r:=r-1;
                IF r#0 THEN ReadWord(b,wb); END
            END
        END;
        END;
        (* Copia de cabos *)
        WHILE r>0 DO
            WriteWord(c,wb); r:=r-1;
            IF r#0 THEN ReadWord(b,wb); END
        END;
        WHILE q>0 DO
            WriteWord(c,wa); q:=q-1;
            IF q#0 THEN ReadWord(a,wa); END
        END;
    UNTIL m=0;
END mezcla;
```

Clasificación por Mezcla Directa (Programa principal)

```
MODULE Directa;  
FROM InOut IMPORT WriteInt, WriteString, WriteLn,  
ReadInt;  
FROM FileSystem IMPORT File,Close,  
Reset,WriteWord, ReadWord, Create;  
  
BEGIN  
  p:=1;  
  longc:=0; (*contador del numero de elementos de c*)  
  REPEAT  
    Separa(c,a,b,p,longc);  
    mezcla(a,b,c,p,longc);  
    p:=2*p;  
  UNTIL p>=longc;  
  (* Fin de ordenación *)  
  Close(c);  
  Close(a);  
  Close(b);  
END Directa.
```


Análisis de clasificación por mezcla directa

Movimientos:



Comparaciones: menor que M , ya que no hay comparaciones en las operaciones de copiado de cabos.

$$C \leq M$$

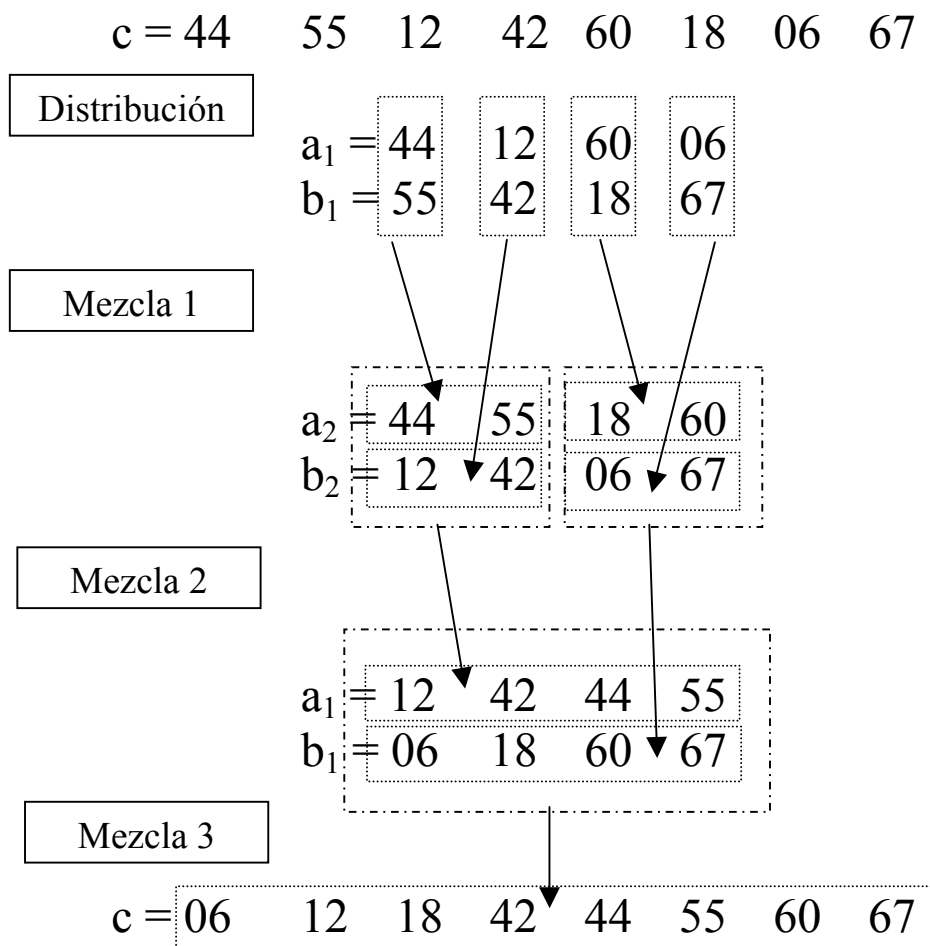
- En las mezclas el número de comparaciones es de poco interés práctico frente a la penalización de tiempo que supone el realizar movimientos.
- La complejidad es parecida a la de los algoritmos avanzados de clasificación en memoria principal.

Inconveniente del uso de este algoritmo en Memoria Principal:

- Almacenar memoria adicional para n elementos (rara vez se utiliza con datos en memoria principal).

Mezcla Directa de una Fase (Mezcla directa balanceada)

- Mejora el método de mezcla directa
- Las fases de división de la mezcla directa pueden eliminarse combinando la fase de división con la de mezcla
- En vez de mezclar en una sola cinta, a, que posteriormente será dividida, puede irse directamente separando en dos, de manera que en el siguiente pase ya estarán divididas (se elimina la distribución).



Movimientos: $M \approx n * \lceil \log_2 n \rceil$

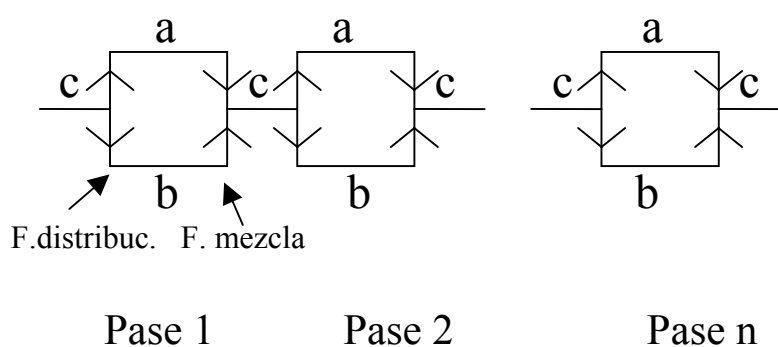
Clasificación por Mezcla Natural

- El algoritmo de Mezcla Directa no tiene en cuenta la existencia de subsecuencias ordenadas a cada paso de dividir la secuencia fuente.

- **Subsecuencia Ordenada:** máximo conjunto de elementos consecutivos a_i, \dots, a_j , de la secuencia a ordenar, que satisfacen:

$$(a_{i-1} > a_i) \text{ Y } (a_k \leq a_{k+1}, \forall k \text{ t.q. } i \leq k \leq j) \text{ Y } (a_j > a_{j+1})$$

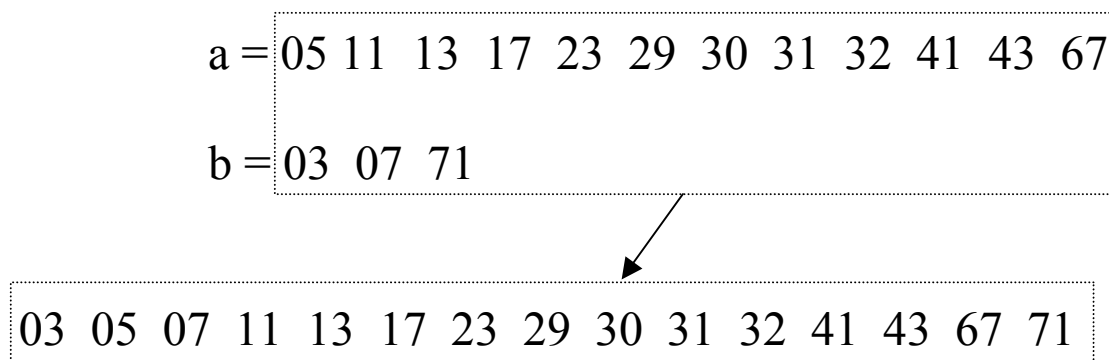
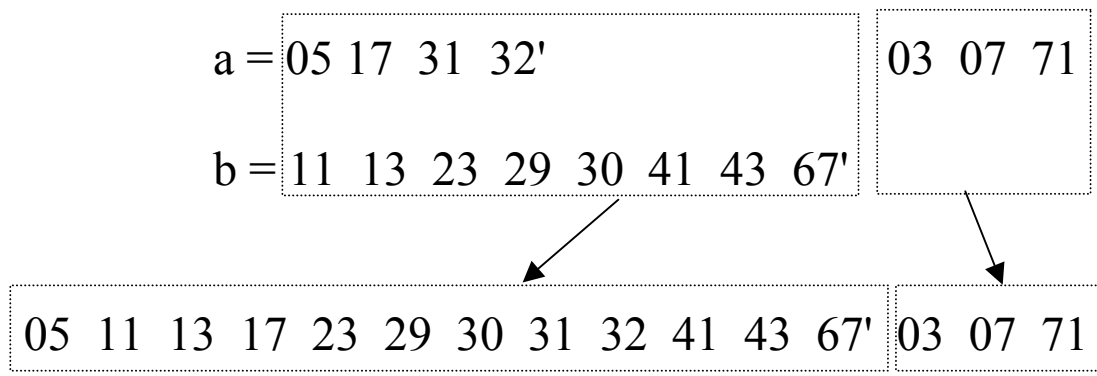
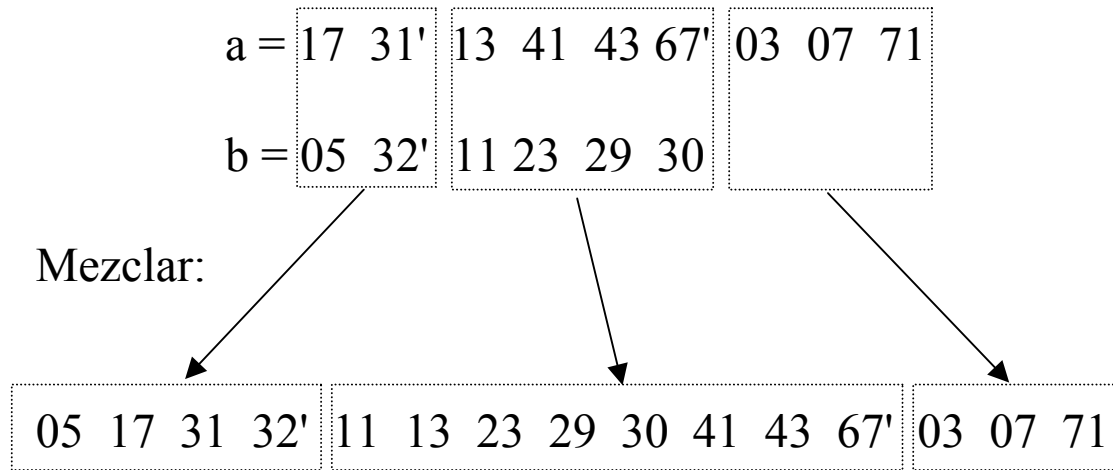
- Si **c**: secuencia inicial de elementos; **a** y **b**: variables de secuencia auxiliar, las fases y pases de la clasificación son:



Cómo opera el algoritmo:

c=17 31' 05 32' 13 41 43 67' 11 23 29 30' 03 07 71

Distribuir alternativamente:



Algoritmo del Método de Mezcla Natural

***PROCEDURE copiado(VAR x,y:File; VAR eor:BOOLEAN;
wx1:INTEGER; VAR wx2:INTEGER);***

(* Copia el valor de x en y. En wx1 recibe el valor a escribir en y.
En wx2 devuelve el valor siguiente de x. Si eor es TRUE
entonces se ha llegado al final de una subsecuencia ordenada de
x*)

BEGIN

WriteWord(y,wx1);

ReadWord(x,wx2);

eor:=(x.eof) OR (wx2<wx1);

END copiado;

PROCEDURE copiar (VAR x,y: File;VAR wx1: INTEGER);

(*Copia en y una subsecuencia ordenada de x. En wx1 devuelve
el ultimo valor leído de x*)

VAR

wx2:INTEGER;

eor:BOOLEAN; (*TRUE: Detectado final de subsecuencia
ordenada*)

BEGIN (*de x hacia y*)

REPEAT copiado(x,y,eor,wx1,wx2);

wx1:=wx2;

UNTIL eor;

END copiar;

```

PROCEDURE distribuir (VAR a,b,c: File);
VAR  wc1: INTEGER;  (*elemento de c que se copia en a o b*)
BEGIN
  Create(a,'nuevoa.dat'); Create(b,'nuevob.dat'); Reset(c);
  ReadWord(c,wc1);
  REPEAT copiar(c,a,wc1);
    IF ~c.eof THEN copiar(c,b,wc1);END;
  UNTIL c.eof;
END distribuir;

```

```

PROCEDURE mezclar(VAR L: INTEGER; VAR a,b,c: File);
(*mezcla a,b en c*)

```

```

VAR wa1,wa2,wb1,wb2: INTEGER;
    eor: BOOLEAN;

```

```

BEGIN

```

```

  Reset(a); Reset(b); Reset(c);

```

```

  L:=0;

```

```

  ReadWord(a,wa1); ReadWord(b,wb1);

```

```

  REPEAT

```

```

    LOOP

```

```

      IF wa1<wb1 THEN

```

```

        copiado(a,c,eor,wa1,wa2);

```

```

        wa1:=wa2;

```

```

        IF eor THEN copiar(b,c,wb1); EXIT; END;

```

```

      ELSE

```

```

        copiado(b,c,eor,wb1,wb2);

```

```

        wb1:=wb2;

```

```

        IF eor THEN copiar(a,c,wa1); EXIT; END;

```

```

      END;

```

```

    END;

```

```

    L:=L+1

```

```

  UNTIL a.eof OR b.eof;

```

```

  (* Copias de cabos finales *)

```

```

    WHILE ~a.eof DO copiar(a,c,wa1); L:=L+1 END;

```

```

    WHILE ~b.eof DO copiar(b,c,wb1); L:=L+1 END;

```

```

END mezclar;

```

cabo subsec. máx. b → c

cabo subsec. máx. a → c

cabo de a → c

cabo de b → c

(Programa Principal)

```
MODULE MezclaNatural;  
FROM InOut IMPORT WriteInt,WriteString,WriteLn,ReadInt;  
FROM FileSystem IMPORT File, Close, Reset, WriteWord,  
ReadWord, Create;  
  
VAR  L:  INTEGER;  (*nº de subsecuencias ordenadas  
mezcladas*)  
      w: INTEGER;  
      a,b,c: File;  
      longc,i,p:  CARDINAL;(* long. de las subsec. por  
combinar*)  
  
BEGIN  
  Reset(c);  
  REPEAT  
    distribuir(a,b,c);  
    mezclar(L,a,b,c);  
  UNTIL L=1;  (* L=1: Fin de la clasificacion *)  
  Close(a); Close(b); Close(c);  
END MezclaNatural.
```

Análisis del método de Mezcla Natural

- El n° total de subsecuencias máximas se divide a la mitad en cada paso. El n° total de movimientos requeridos es:

$$M \leq 2n * \lceil \log_2 n' \rceil$$

donde $n' = N^\circ$ de subsecuencias máximas de la secuencia original.

- El peor caso se da cuando $n' = n$, es decir, cuando la secuencia original está ordenada en forma inversa (en este caso particular, la forma de operar es equivalente al método de mezcla directa)
- El n° de comparaciones que se necesitan es mayor porque, además de las necesarias para seleccionar elementos, se requieren otras para determinar el final de cada subsecuencia máxima.
- Se aplicará el caso “*menor que*” en la expresión anterior, si como resultado de una distribución, dos o más subsecuencias procedentes de la cinta fuente se pueden agrupar en una sola. Ejemplo:

$c = 17 \ 19' \ 13 \ 57' \ 23 \ 29' \ 11 \ 59' \ 31 \ 37' \ 07 \ 61' \ 41 \ 43$

$a = \boxed{17 \ 19 \ 23 \ 29 \ 31 \ 37 \ 41 \ 43}$

$b = 13 \ 57' \ 11 \ 59' \ 07 \ 61$

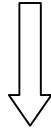
Mezcla Balanceada Múltiple

- Persigue reducir el número de movimientos \Rightarrow Reducir el número de pases
- Se consigue realizando la distribución entre dos o más cintas (en la mezcla se combinan varias, mas de dos, subsecuencias ordenadas)
- Además, se utilizará una mezcla en una sola fase (se elimina la fase de distribución, salvo la inicial)
- Distribuir las subsecuencias en más de dos secuencias (mezcla múltiple), para reducir el nº de pasos.
- Se requiere un número par de secuencias de tal forma que, en un pase, la mitad de ellas actuarán como fuentes y la otra mitad como destino y, en el pase siguiente, invertirán sus papeles.

¿Cómo opera el algoritmo?

f0 = 50' 40' 38' 35' 30' 25' 20 21' 10 41' 15 31' 5 60

Distribución



f1 = 50' 35' 20 21' 05 60

f4 = 38 40 50' 05 60

f2 = 40' 30' 10 41

f5 = 25 30 35

f3 = 38' 25' 15 31

f6 = 10 15 20 21 31 41

Mezcla



**f1 = 10 15 20 21 25 30
31 35 38 40 41 50**

f4 = 38 40 50' 05 60

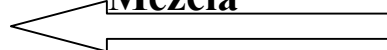
f2 = 05 60

f5 = 25 30 35

f3 =

f6 = 10 15 20 21 31 41

Mezcla



**f1 = 10 15 20 21 25 30
31 35 38 40 41 50**

**f4 = 05 10 15 20 21 25
30 31 35 38 40 41
50 60**

f2 = 05 60

f5 =

f3 =

f6 =

Mezcla



- Estructura de datos utilizada para las cintas: arreglo secuencias (archivos)
- Tipos y variables más significativos:

type seqno = [1.. N]; (** el índice N, n° par, es el n° de secuencias o cintas**)

t : ARRAY seqno OF seqno; (**mapa de índice de cintas, es decir, se referencian no como f[i] sino como f[t[i]]**)

fini : Sequence; (**secuencia inicial de elementos**)

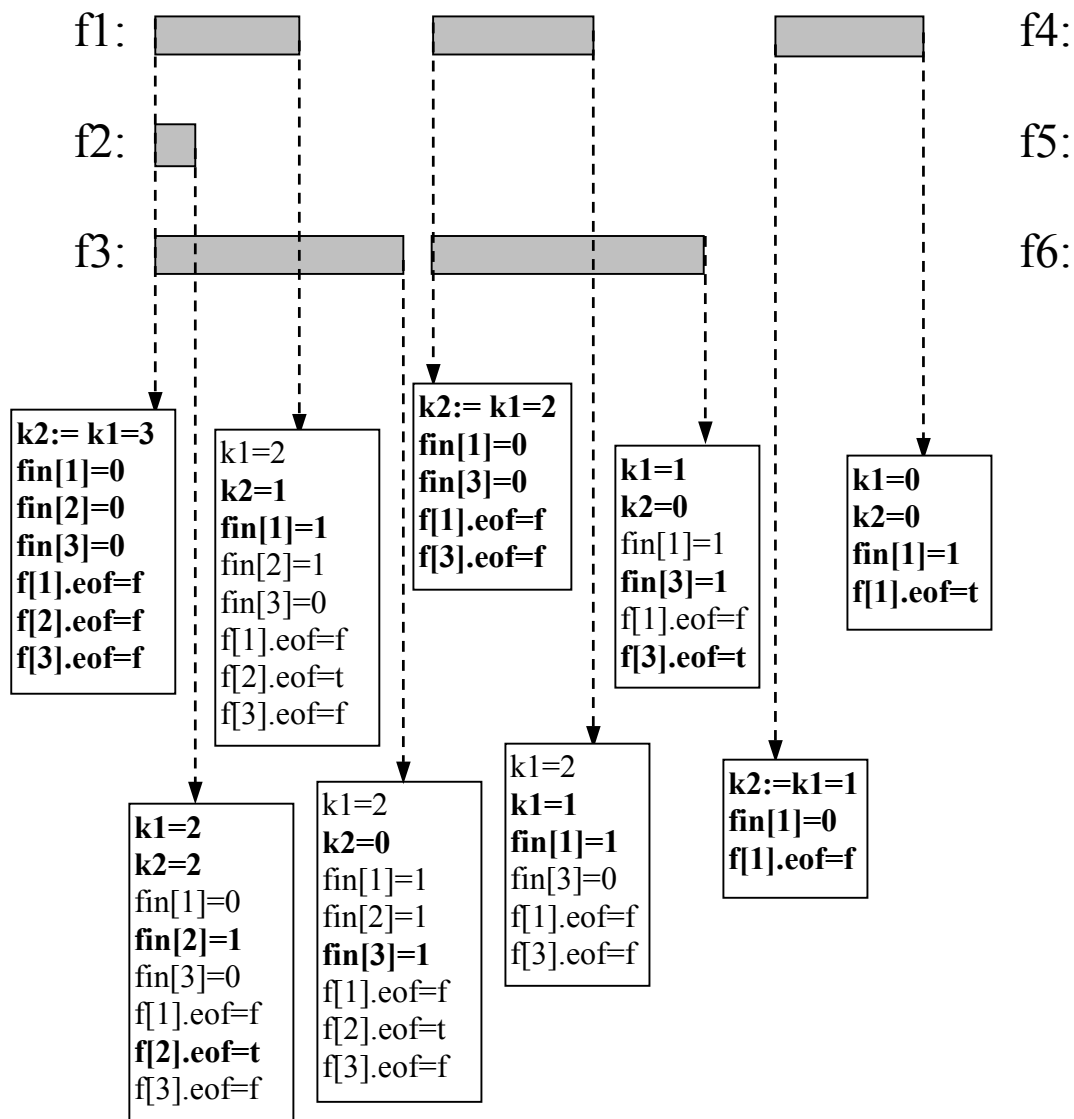
f : ARRAY seqno OF Sequence; (**N cintas en el proceso de clasificación: N/2 de entrada y N/2 de salida**). Nota: Si $f[t[i]].eof = TRUE$ ya no se considerará más la cinta $f[t[i]]$ en todo el proceso que implique una mezcla determinada.

fin: ARRAY seqno OF INTEGER; (**si $fin[t[i]] = 1$ se ha alcanzado eor en una subsecuencia de la cinta $f[t[i]]$. Si no hay eof simultáneo, dicha cinta no se volverá a considerar hasta cambiar la cinta fuente en el proceso que implica una mezcla determinada**).

k1: denota el n° real de cintas de entrada que se están utilizando. Un decremento de k1 denota que una cinta fuente ha llegado al final: " $f_i.eof = true$ ".

k2: denota el n° de cintas disponible en cada momento para seleccionar el siguiente elemento. Un decremento de k2 denota que se ha llegado al final de subsecuencia en una cinta fuente.

Ejemplo¹ de manejo de *variables e índices significativos* en la implementación del **algoritmo de mezcla balanceada múltiple**



¹ La distribución asimétrica de subsecuencias en las tres cintas es posible porque puede darse el caso de que dos o más subsecuencias, procedentes de sus mezclas respectivas, al caer en la cinta correspondiente, puedan constituir una única subsecuencia.

Esquema del Algoritmo:

MODULE MezclaBalanceada;

VAR i, j: seqno;

L: INTEGER;(*nº de subsec. totales que han de distribuirse*)

t: ARRAY seqno OF seqno;

BEGIN (*distribuir subsecuencias iniciales a t[1]... t[Nh]*)

FOR i:=1 TO Nh DO Reset(f[i]) END;

Reset(fini); j := Nh; L := 0; (* Nh = N/2 *)

REPEAT

IF j < Nh THEN j := j+1 ELSE j := 1 END;

copiar una subsecuencia de f_{ini} a la cinta j-ésima

L := L+1 (*nº de subsecuencias distribuidas*)

UNTIL fini.eof;

FOR i:=1 TO Nh DO fin[i]:=0; END;

FOR i := 1 TO N DO t[i] := i END;(*inicializa mapa de
índices*)

REPEAT (*mezclar de t[1]...t[Nh] a t[Nh+1]... t[N]*)

restablecer cintas de entrada;

L := 0;

j := Nh+1; (*j = índice de la cinta de salida*)

REPEAT

L := L+1; k2:=k1;

mezclar una subsecuencia de cada entrada a t[j]

IF j < N THEN j := j+1 ELSE j := Nh+1 END

UNTIL agotar todas las entradas;

Intercambiar cintas origen ↔ cintas destino;

UNTIL L = 1

(*finalmente la secuencia clasificada es t[1]*)

END MezclaBalanceada;

(1) *copiar una subsec. máx. de f_{ini} a la secuencia j ;*

```
REPEAT copiado(fini,f[j],eor, wfini1, wfini2);  
IF ~fini.eof THEN wfini1:=wfini2; END;  
UNTIL eor;
```

```
PROCEDURE copiado(VAR x,y:File; VAR eor:BOOLEAN;  
wx1:INTEGER; VAR wx2:INTEGER);
```

(* Copia el valor de x en y; lee el primero de x y determina si es fin de subsecuencia *)

```
BEGIN  
  WriteWord(y,wx1);  
  ReadWord(x,wx2);  
  eor:=(x.eof) OR (wx2<wx1);  
END copiado;
```

(2) *restablecer cintas de entrada;*

k1: denota el *nº real de cintas de entrada* que se están utilizando. Un decremento de k1 denota que una cinta fuente ha llegado al final: " $f_i.eof=true$ ".

```
IF L < Nh THEN k1 := L ELSE k1:= Nh END;  
FOR i := 1 TO k1 DO Reset (f[t[i]]); END;
```

Lee primer elemento de cada cinta (*ver código en libro*)

(3) *mezclar una subsecuencia de cada entrada a $t[j]$*

- Selección repetida de las subsec. de las k1 cintas fuentes disponibles y transporte de la subsec. mezcla a la cinta de salida actual.

- **k2:** denota el *nº de cintas disponible* en cada momento para seleccionar el siguiente elemento. Un decremento de k2 denota que se ha llegado al final de subsecuencia en una cinta fuente.

- **t:** mapa de cintas auxiliar que nos indica los índices de las secuencias disponibles.

REPEAT

*seleccionar la llave mínima (min), sea t[mx] el n° de cinta en que ocurre; (*ver código en el libro*)*

copiado (f[t[mx]], f[t[j]], eor, min, wx2);

wx[t[mx]]:=wx2;(*almacena elemento leído en copiado*)

IF eor THEN fin[t[mx]]:=1; END;

IF f[t[mx]].eof THEN (*eliminar cinta t[mx]*)

 (* decrementar k1 y k2*)

 k1:=k1-1;k2:=k2-1;

ELSEIF eor THEN (*terminar subsec. en f[t[mx]]*)

 k2:=k2-1

END;

UNTIL k2 = 0

(4) *agotar todas las entradas:*

IF j<N THEN j:=j+1 ELSE j:=Nh+1 END;

(*Poner a cero el indicador de fin de subsecuencia*)

FOR i:=1 TO N DO fin[t[i]]:=0; END;

UNTIL k1 = 0

(5) *Intercambiar cintas origen \leftrightarrow cintas destino;*

FOR i:= 1 to Nh DO

 tx:=t[i]; t[i]:=t[i+Nh]; t[i+Nh]=tx;

END;

(Con estas indicaciones ver algoritmo de págs.111-114 del libro)

Análisis del Método de Mezcla Balanceada

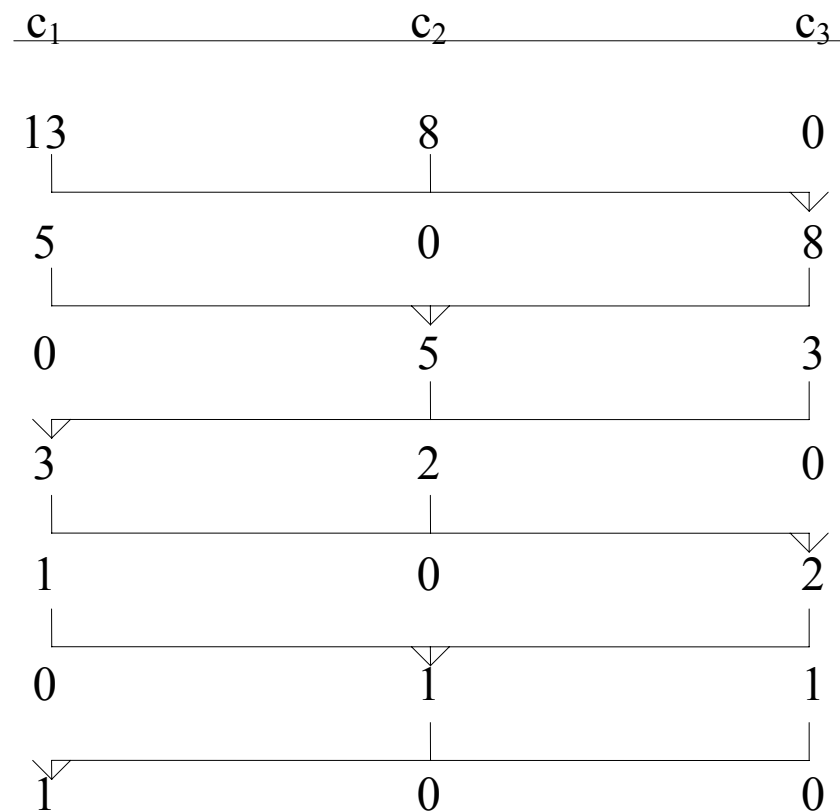
- Siendo N el nº de cintas destino, n el de elementos y n' el de subsecuencias máximas de la secuencia inicial, el nº de movimientos necesarios viene dado por:

$$M \leq n * \lceil \lg_{(N/2)} n' \rceil$$

- Peor caso: $n'=n$ (secuencia original ordenada e invertida)
- Como en el caso de la mezcla natural, se aplicará el caso “*menor que*” en la expresión anterior, si como resultado de una mezcla-distribución, dos o más subsecuencias, existentes en una cinta destino y procedentes de la mezcla de las cintas fuente, se pueden agrupar en una sola.

Clasificación Polifásica

- Al igual que la mezcla balanceada, la Clasificación Polifásica une en una sola fase las operaciones de distribución y mezcla.
- Mejora: abandonar la noción de pases estrictos ($N/2$ cintas fuentes y $N/2$ cintas destino).
- Ejemplo: Clasificación polifásica de 21 subsecuencias con 3 cintas:



(Cada vez que se agota una de las secuencias fuente, se convierte en el destino)

➤ No siempre es satisfactoria en todos los casos. Ejemplo:

c_1	c_2	c_3
14	8	0
6	0	8
0	6	2
2	4	0
0	2	2
2	0	0

Distribución de subsecuencias iniciales:

- ¿Qué distribución de subsecuencias iniciales lleva a un funcionamiento adecuado?:
- Procedemos en sentido contrario, comenzando por la última línea de la distribución.

- Ejemplo: Construcción de la clasificación polifásica satisfactoria de tres cintas:

Nivel	$c_1(n)$	$c_2(n)$	$c_1(n)+c_2(n)$
0	1	0	1
1	1	1	2
2	2	1	3
3	3	2	5
4	5	3	8
5	8	5	13
6	13	8	21
7	21	13	34

- Para $n=0$

$$c_1(0)=1$$

$$c_2(0)=0$$

- Para cada nivel ($n>0$):

$$c_2(n+1) = c_1(n)$$

$$c_1(n+1) = c_1(n) + c_2(n) = c_1(n) + c_1(n-1)$$

- Se cumple que $c_1(n)$ son siempre números de Fibonacci y $c_2(n)$ siempre es el predecesor de $c_1(n)$.

- **Conclusión**: La clasificación polifásica con 3 cintas será satisfactoria si la distribución inicial de subsecuencias ordenadas entre las cintas fuentes es tal que son números consecutivos de Fibonacci.

Ejemplo que demuestra que "la suma en el primer nivel debe ser un número de Fibonacci" no es condición suficiente:

c_1	c_2	c_3
22	12	0
10	0	12
0	10	2
2	8	0
0	6	2
2	4	0
0	2	2
2	0	0

Ejemplo que demuestra que la conclusión de la página anterior no es condición necesaria. Es decir, existen otras distribuciones iniciales para las que la clasificación es satisfactoria:

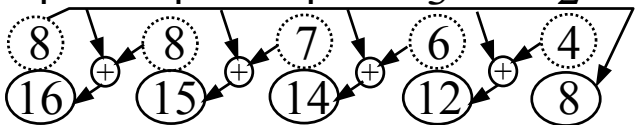
c_1	c_2	c_3
11	7	0
7	0	4
3	4	0
0	1	3
1	0	2
0	1	1
1	0	0

- **Ejemplo** de Clasificación polifásica de 65 subsecuencias con 6 cintas:

f_1	f_2	f_3	f_4	f_5	f_6
16	15	14	12	8	
8	7	6	4	0	8
4	3	2	0	4	4
2	1	0	2	2	2
1	0	1	1	1	1
0	1	0	0	0	0

Construcción de la clasificación polifásica de seis cintas satisfactoria:

Nivel	$c_1(n)$	$c_2(n)$	$c_3(n)$	$c_4(n)$	$c_5(n)$	Suma $c_i(n)$
0	1	0	0	0	0	1
1	1	1	1	1	1	5
2	2	2	2	2	1	9
3	4	4	4	3	2	17
4	8	8	7	6	4	33
5	16	15	14	12	8	65



$$c_5(n+1) = c_1(n)$$

$$c_4(n+1) = c_1(n) + c_5(n) = c_1(n) + c_1(n-1)$$

$$c_3(n+1) = c_1(n) + c_4(n) = c_1(n) + c_1(n-1) + c_1(n-2)$$

$$c_2(n+1) = c_1(n) + c_3(n) = c_1(n) + c_1(n-1) + c_1(n-2) + c_1(n-3)$$

$$c_1(n+1) = c_1(n) + c_2(n) = c_1(n) + c_1(n-1) + c_1(n-2) + c_1(n-3) + c_1(n-4)$$

con $c_1(0) = 1$ y $c_i(0) = 0$ para $i=2..5$

donde $c_1(n)$ son los n° de Fibonacci de orden 4.

Clasif. Polifásica: Generalizando, para N cintas se toman $N-1$ números consecutivos $[a_i + a_{i+1} + \dots + a_{i+(p-2)}]$ de la sucesión de Fibonacci² de orden $N-2$

La distribución inicial en cada una de las cintas es

$$\begin{aligned} C_1 &= a_i + a_{i+1} + \dots + a_{i+(p-2)} \\ C_2 &= a_{i+1} + a_{i+2} + \dots + a_{i+(p-2)} \\ &\dots \\ C_{N-2} &= a_{i+(p-3)} + a_{i+(p-2)} \\ C_{N-1} &= a_{i+(p-2)} \end{aligned}$$

Donde

$$a_1, a_2, \dots, a_{i-1}, a_i, a_{i+1}, \dots, a_{i+(p-2)}, a_{i+(p-1)}, a_{i+p}, a_{i+(p+1)}, \dots, a_n$$

$\begin{array}{c} \uparrow \qquad \qquad \uparrow \\ \boxed{\text{N-1 términos}} \end{array}$

es la sucesión de Fibonacci de orden $N-2$

- Si no es posible obtener un intervalo de $N-1$ números consecutivos tal que $\sum_1^{N-1} C_i = \text{No. inicial de subsec.}$, entonces se escoge el intervalo $[a_j, \dots, a_{j+(p-2)}]$ que cumpla:

$$[1] \sum_1^{N-1} C_i > \text{No. inicial de subsec.}$$

$$[2] \text{ Para } [a_{j-1}, \dots, a_{j+(p-1)}] \Rightarrow \sum_1^{N-1} C_i < \text{No. inicial de subsec.}$$

Finalmente, se recurre a tantas subsecuencias ficticias (vacías) como sean necesarias para completar las subsecuencias que faltan.

² La sucesión de Fibonacci de orden n es aquella en la que cada término se obtiene como suma de los $(n+1)$ anteriores términos de la sucesión.

Ejemplo Clasif. Polifásica: Cálculo de la distribución de subsecuencias para un problema de 6 cintas (N=6) y 233 subsecuencias.

La sucesión de Fibonacci de orden 4 (N-2) es:

$$1, 1, 2, 4, 8, 16, 31, 61, \dots$$

Hay que escoger un intervalo de 5 (N-1) números consecutivos de la sucesión. Sea:

$$[a_1, a_2, a_3, a_4, a_5] = [1, 2, 4, 8, 16]$$

entonces

$$C_1 = 1+2+4+8+16 = 31$$

$$C_2 = 2+4+8+16 = 30$$

$$C_3 = 4+8+16 = 28$$

$$C_4 = 8+16 = 24$$

$$C_5 = 16$$

Pero $\sum_1^5 C_i = 107 < N^{\circ} \text{ subsec.}$

Por tanto, habrá que escoger el siguiente intervalo de números consecutivos de la sucesión, es decir,

$$[a_1, a_2, a_3, a_4, a_5] = [2, 4, 8, 16, 31]$$

entonces

$$C_1 = 2+4+8+16+31 = 61$$

$$C_2 = 4+8+16+31 = 59$$

$$C_3 = 8+16+31 = 55$$

$$C_4 = 16+31 = 47$$

$$C_5 = 31$$

Y como $\sum_1^5 C_i = 253 > N^{\circ} \text{ subsec.}$

El resto de 20 ($n'=253-233$) subsecuencias se completan con subsecuencias ficticias.

Existen "nº de subsecuencias totales" para los que no se satisfacen las condiciones anteriores. En este caso simulamos la existencia de **subsecuencias ficticias** (vacías) que añadimos a las originales hasta alcanzar un número de subsecuencias que si satisfagan dichas condiciones. Interesa distribuir uniformemente las subsecuencias ficticias.

- **La clasificación polifásica precisa del control del número exacto de subsecuencias ordenadas en cada cinta:** se necesita conservar las llaves del último elemento de la subsecuencia ordenada para evitar el caso en que dos subsecuencias lleguen al mismo destino y constituyan una sola subsecuencia ordenada (alteraría el nº final de subsecuencias en esa cinta).

Algoritmo de distribución de un nº de subsecuencias en N cintas:

1. Sea la meta de distribución el nº de subsecuencias para el nivel 1.
2. Distribuir conforme a la meta establecida.
3. Si se alcanza la meta, se calcula el nº de subsecuencias de cada cinta para el siguiente nivel; las diferencias entre los de este nivel y los del anterior constituyen la nueva meta de distribución. Volver al paso 2. Si la meta no se puede lograr, porque se ha agotado la fuente, finalizar la distribución.

Tratamiento de subsecuencias ficticias en la distribución inicial:

- En el paso 2 tenemos que tener en cuenta las subsecuencias ficticias.

1. Al pasar de un nivel a otro, se registra la siguiente meta por las diferencias s_i para $i = 1 \dots N-1$ (s_i denota el nº de subsecuencias por poner en la cinta i en este paso).

2. Se suponen s_i subsecuencias ficticias en la cinta i .

3. Se considera la distribución siguiente como la sustitución de cada subsecuencia ficticia por las reales; registrando cada sustitución mediante la resta de 1 a s_i .

4. Al final, s_i indicará el nº de subsecuencias ficticia en la secuencia i cuando la fuente original queda vacía.

Tratamiento de las subsecuencias ficticias en la mezcla:

1. Si se selecciona una de ellas de una cinta i , significa que la cinta i se ignora durante esta combinación, dando como resultado una mezcla procedente de menos de $N-1$ fuentes.

2. Si la mezcla se produce entre $N-1$ subsecuencias ficticias, en la cinta de salida se registrará una subsecuencia ficticia.

Notas sobre la implementación:

Es muy parecida a la de la mezcla balanceada con las siguientes diferencias:

- (1) En vez de $N/2$ cintas destino sólo hay una en cada pase.
- (2) En vez de intercambiar las $N/2$ cintas iniciales con las $N/2$ destino al finalizar cada pase, las cintas se van rotando. Esto se consigue con el indexado de cintas t , es decir, $f[t[i]]$.
- (3) Hay que tener en cuenta el tratamiento de las subsecuencias ficticias indicado en la transparencia anterior.
- (4) Cuando existe al menos un $s_i = 0$ indica que empiezan a mezclarse subsecuencias no ficticias. Se denotará por k el número de subsecuencias reales que participan en una mezcla.
- (5) La determinación de la finalización de una fase se hace a partir de los coeficientes c_j . Estos se calcularon en la fase de distribución inicial y ahora se vuelven a recalcular en sentido inverso.

(Con todas estas indicaciones, ver el código del programa que aparece en las pgs. 125-128 del libro de texto)

Clasificación mediante archivos indexados

- Las implementaciones de secuencias en algunos dispositivos de memoria secundaria, tales como discos duros, permiten un acceso cuasialeatorio (**fichero**):

Acceso aleatorio al sector + Offset de acceso secuencial

- El método de clasificación mediante archivos indexados se basa en considerar, asociado a cada llave, la dirección física del registro que caracteriza.
- Se crea un segundo archivo, **archivo de índices**, en el que se almacenan pares (dirección, llave).

(Incluir figura 3.15)

- Las operaciones de clasificación, búsqueda y borrado, realmente, se realiza en el archivo índice. Si además, éste es pequeño, se puede llevar a memoria principal.
- Si se desea ordenar el archivo de datos según distintos criterios (distinto tipo de llave) se creará un archivo de índices para cada uno.

Refinamiento:

1. Fragmentar el archivo de índices en bloques para evitar reordenaciones excesivas al insertar una nueva llave.
2. Agrupar las llaves en bloques con pares (x_i, b_i) donde b_i es la dirección física del bloque en el cual el primer registro tiene los contenidos de la llave de valor x_i .

Estrategia:

1. Procurar que la distribución inicial de registros en cada bloque no sea máxima.
2. Se clasifica el archivo de índices.
3. Para recuperar una llave, x , se busca en el archivo índice la mayor llave menor o igual que x , cuya dirección asociada dará acceso al bloque.
4. Se busca en el bloque dicha llave, que nos dará acceso al registro de datos buscado.

(Insertar fig. 3.16)
(ojo, en el libro está mal)

Estas técnicas, basadas en archivos indexados, dan soporte a gran número de aplicaciones en **Bases de Datos**.

Tablas de Dispersión (HASHING)

Problema:

- ¿Cómo organizar un conjunto de elementos, caracterizados por una llave, cuando el número de llaves posibles es mucho mayor que el de llaves reales a almacenar? Además, la recuperación del elemento deberá realizarse con el "menor esfuerzo" posible.

Ejemplo: Agenda de teléfonos con tabla de 10^3 índices (existen 10^9 números de teléfono posibles).

Soluciones:

- (a) Arreglos + algoritmos de búsqueda.
- (b) Árboles y/o listas + algoritmos de búsqueda.
- (c) Tablas de dispersión: se basan en encontrar una función de transformación apropiada, H , que a partir de una llave, X , nos dé la dirección de memoria, D , del elemento que posee dicha llave:

$$H: X \longrightarrow D$$

- Estas direcciones pueden hacer referencia a memoria primaria (arreglo), pero en general las tablas de dispersión adquieren su máximo potencial cuando se implementan en memoria secundaria.
- **El TDA Tabla de Dispersión**: es un tipo de datos homogéneos, compuesto por un número de bloques fijo, cada una con un número de celdas fijo, a las que se accede mediante la dirección de memoria resultante de aplicar una función de transformación. Sobre esta TDA se definen los operadores *Insertar*, *Buscar* y *Eliminar*.

Otras Definiciones:

Def.1: Se dice que dos llaves distintas x_1 y x_2 son *sinónimas* para una función de transformación $H(x)$ si $H(x_1)=H(x_2)$.

Def.2: Se dice que se ha producido *desbordamiento* cuando una nueva llave se aplica a una dirección de memoria completamente ocupada.

Def.3: Se dice que se ha producido una *colisión* cuando dos llaves distintas se aplican sobre el mismo bloque. En el caso habitual de que un bloque contiene una sola celda, el desbordamiento y la colisión se producen simultáneamente.

Def.4: Se denomina *densidad de llaves* al cociente entre el número de llaves en uso, m , y el número total de llaves posibles, n_x . Se denomina *factor de carga*, α , al cociente entre el nº de llaves en uso y el nº total de registros almacenables:

$$\alpha = m / (s*b)$$

donde s es el nº de celdas por bloque y b el nº de bloques.

Búsqueda

➤ La operación de búsqueda de una llave, x , en una tabla de transformación implica:

1) Calcular su dirección asociada: $d = H(x)$

2) Verificar si $d^{\text{llave}} = x$

➤ Dos cuestiones:

1) ¿Qué tipo de función H debe utilizarse?

2) ¿Cómo resolver el *problema de colisiones* ?

Elección de una Función de Transformación

- Requisitos básicos de una función de transformación:

- 1) Cálculo sencillo y rápido.
- 2) Distribución de llaves uniforme (reduce el nº de colisiones).

➤ Ejemplo 1: **Función Resto de División**

$$H(x) = \text{ORD}(x) \text{ MOD } M$$

siendo

$\text{ORD}(k)$: nº ordinal de la llave x .

M : número de bloques de la tabla.

La elección del valor M puede ser crítica:

Ejemplo: "Las llaves posibles son números acabados en cero". Una elección de $M=10$ supondría que el resto de la división siempre sería cero y, por tanto, se producirían colisiones constantemente.

Ejemplo 2: Función plegado. La dirección se obtiene dividiendo las llaves en partes iguales y sumando todas ellas.

La suma de las partes se puede realizar de dos formas:

Plegado por desplazamiento: consiste en sumar las partes directamente.

Plegado por las fronteras: consiste en plegar el identificador por las fronteras de las partes y sumar los dígitos coincidentes.

Insertar fig. 3.19

Manejo de las Colisiones.

Para poder determinar si existe o no colisión, debe inicializarse primeramente todas las celdas de la tabla a un valor *cte* y que sea diferente a la llave de cualquier elemento que se pueda almacenar.

La secuencia de las direcciones de exploraciones secundarias debe ser siempre igual para una llave determinada.

Se necesita una función $G(i)$ (con $i=0,1,2,\dots$ el número de intentos a realizar) que junto con $H(x)$ calcule cuál será la nueva posición a examinar:

$$\text{Dirección: } H(x) + G(i)$$

- Estrategia de Exploración Lineal: ir buscando *en la posición siguiente*, hasta encontrar el elemento con la llave especificada o una posición vacía.

$$\begin{aligned} d_0 &= H(x) \\ d_i &= (d_0 + i) \text{ MOD } M, \quad i = 1 \dots M-1 \end{aligned}$$

Inconveniente: agrupa las entradas que sufren colisión alrededor de las llaves primarias (pérdida de uniformidad en la distribución)

- Estrategia de Exploración Cuadrática:

$$\begin{aligned} d_0 &= H(x) \\ d_i &= (d_0 + i^2) \text{ MOD } M, \quad i > 0 \end{aligned}$$

Ventaja: evita la agrupación primaria.

Inconveniente: no realiza la búsqueda exhaustiva en toda la tabla, es decir, puede ocurrir que no encontremos una posición libre aunque en realidad queden algunas (si M es número primo de la forma $4p+3$, p entero, garantiza que la

exploración cuadrática recorre todos los bloques de la tabla).

- Técnica de Rehashing

$$d_0 = H(x)$$
$$d_i = (d_0 + f_i) \text{ MOD } M, \quad i = 1, \dots, m$$

donde cada f_i es una función de dispersión

- Estrategia de Encadenamiento Directo: asociar todos los elementos con dirección primaria idéntica $H(x)$ en una lista ligada. Los elementos de la lista pueden estar en la tabla primaria o no; en el segundo caso, al almacenamiento donde están asignados, se llama *área de desbordamiento*.

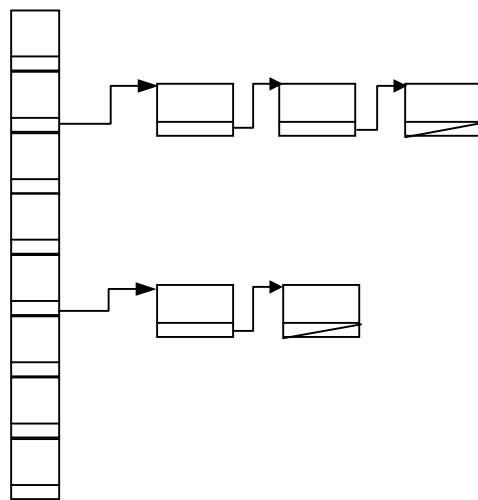


Fig.1 Con área de desbordamiento

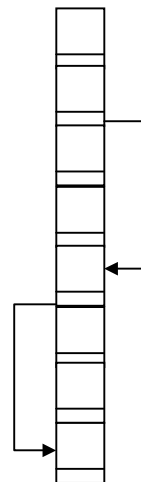


Fig.2 Sin área de desbordamiento

Inconvenientes:

- (1) Mantener la lista.
- (2) Reservar para cada elemento un apuntador (índice) a su lista de elementos en colisión.

Algoritmo de búsqueda:

```
d := H(x); i:= 0; (* se obtiene el índice a partir de la llave x*)
REPEAT
  IF d^.llave = x THEN elemento encontrado
  ELSEIF d^.llave = free THEN elemento no está en T
  ELSE  (*hay colisión*)
    i := i+1; d := H(x) +G(i)  (*genera una direc. alternativa*)
  END
UNTIL encontrado o no está en la tabla (o tabla llena)
```

Análisis de las Tablas de Transformación

Si **n**: tamaño de la tabla

E: N° de exploraciones para recuperar o insertar una llave

Mejor de los casos. No se producen colisiones y el número de celdas por bloque es pequeño.

$$E \neq f(n)$$

Caso promedio: todas las llaves poseen la misma probabilidad y la función H las distribuye uniformemente sobre la tabla.

La probabilidad, p_1 , de encontrar un bloque vacío en la primera comparación será:

$$p_1 = (n-k)/n \quad \text{siendo } k \text{ el n° posiciones ocupadas}$$

La probabilidad, p_2 , de que se necesite una segunda comparación es igual a la probabilidad de que se haya producido desbordamiento en la primera por la probabilidad condicionada de hallar libre el siguiente bloque que se consulta habiéndose producido desbordamiento en el primer intento:

$$p_2 = [k/n] * [(n-k)/(n-1)]$$

Generalizando, la probabilidad, p_i , de que una inserción requiera i exploraciones:

$$p_i = [k/n] * [(k-1)/(n-1)] * [(k-2)/(n-2)] * \dots * [(n-k)/(n-i+1)]$$

Puede demostrarse que el **valor esperado** de exploraciones requeridas, después de insertar la llave $(k+1)$ -ésima, es:

$$E_{k+1} = \sum_{i=1}^{k+1} i \cdot p_i = \frac{n+1}{n-k+1}$$

Suponiendo que n es el tamaño de la tabla y m el nº de llaves almacenadas en ella, el nº promedio de exploraciones, E , para recuperar una llave aleatoria de dicha tabla es:

$$E = \frac{\sum_{k=1}^m E_k}{m} = (n+1) \frac{\sum_{k=1}^m \frac{1}{(n-k+2)}}{m} = (n+1) \frac{(H_{n+1} - H_{n-m+1})}{m}$$

Sabiendo que, aproximadamente, $H_n \approx (\ln n + \gamma)$ y llamando $\alpha = m/n$, la expresión anterior se puede simplificar a:

$$E \approx -\frac{\ln(1-\alpha)}{\alpha}$$

Aún cuando la tabla esté llena en un 90%, apenas se necesitan 2,56 exploraciones, en término medio, para localizar una llave o una posición vacía.

Para la estrategia de encadenamiento (Caso Promedio). Puede demostrarse que:

$$E \approx 1 + \frac{\alpha}{2}$$

Hashing frente a asignación dinámica

- Ventajas:

El rendimiento relativo a los pasos de comparación para la recuperación e inserción es superior al de la organización de árbol más depurada.

- Desventajas:

- 1) Tamaño de tabla fijo: es necesaria la estimación a priori del n° de elementos.
- 2) La tarea de eliminación de llaves de la tabla es más costosa que la de inserción o la de búsqueda. Solución: encadenamiento directo en un área de desbordamiento (manejo de listas, asignación dinámica).
- 3) La implementación de otras funciones típicas sobre estas tablas, tales como encontrar el máximo, imprimir la tabla ordenadamente, etc. son muy costosas.
- 4) Su organización de datos no lleva asociada el concepto de ordenación: la inserción de una llave en la tabla es sólo función de las funciones $H(x)$ y $G(i)$ y, éstas, no son precisamente funciones de ordenación.