Árboles B

Tema 7

1. DEFINICION DE LOS TIPOS DE DATOS

```
#define   m      4      /* Orden del árbol. */
#define   Nulo   -1     /* Puntero nulo */
#define   nK     K[0]   /* Macro para acceder al número de claves. */

#define   NPILA  16     /* Tamaño de la pila. */

typedef int  TClave;    /* Tipo de las claves. */
typedef long TPuntero;  /* Tipo de los punteros dentro de los nodos */

typedef int  TPosicion; /* Tipo de las posiciones dentro de los arrays. */

typedef struct
{ TClave K[m];          /* Array de claves. En K[0] estará n. */
  TPuntero A[m];        /* Array de punteros a nodos. */
} TNodoFich;

typedef struct
{ TPuntero Direccion;   /* Direccion del nodo original en fichero. */
  TClave K[m + 1];      /* Idem a TNodoFich, pero permite sobrecarga. */
  TPuntero A[m + 1];    /* Idem a TNodoFich, pero permite sobrecarga. */
} TNodoMem;

typedef struct
{ TNodoMem Nodo;
  TPosicion Posicion;   /* Posición por la que se desciende. */
} TPila[NPILA];
```

2. ALGORITMOS PARA LEER/ESCRIBIR NODOS

```
void LeerNodo(TPuntero Dir, TNodoMem *NodoM)
/* Dir: dirección en fichero donde se encuentra el nodo a leer (E). */
   NodoM: dirección del nodo en memoria donde se meten los datos (S). */
{
  TNodoFich NodoF;
  lseek(fd, Dir, SEEK_SET);
  read(fd, &NodoF, sizeof(TNodoFich));
  NodoM->A[0] = NodoF.A[0];
  for (i = 1; i <= NodoF.nK; i++)
     { NodoM->K[i] = NodoF.K[i];
       NodoM->A[i] = NodoF.A[i];
     }
  NodoM->nK = NodoF.nK;
  NodoM->Direccion = Dir;
}

void EscribirNodo(TNodoMem *NodoM)
/* NodoM: dirección en memoria del nodo que se va a escribir. */
{
  TNodoFich NodoF;
  NodoF.A[0] = NodoM->A[0];
  for (i = 1; i <= NodoM->nK; i++)
     { NodoF.K[i] = NodoM->AK[i];
       NodoF.A[i] = NodoM->A[i];
     }
  NodoF.nK = NodoM->nK;
  lseek(fd, NodoM->Direccion, SEEK_SET);
  write(fd, &NodoF, sizeof(TNodoFich));
}
```

3. ALGORITMO DE INSERCIÓN DEL PAR (CLAVE, PUNTERO) EN UN NODO

```
Procedimiento InsertarClaveYPuntero(Nodo, Pos, Clave, Puntero)
Nodo: donde se va a insertar la clave y el puntero (E/S).
Pos: posición dentro del nodo donde se insertará la clave (E).
Clave: a insertar (E).
Puntero: a insertar (E).
{
  for (i = Nodo.nK; i >= Pos; i--)
     { Nodo.A[i + 1] = Nodo.A[i];
       Nodo.K[i + 1] = Nodo.K[i];
     }

  Nodo.K[Pos] = Clave;
  Nodo.A[Pos] = Puntero;

  Nodo.nK++;
  return;
}
```

4. ALGORITMO DE EXTRACCION DE UNA CLAVE EN UN NODO

```
Procedimiento ExtraerClave(Nodo, Pos)
Nodo: del cual se va a extraer la clave (E/S).
Pos: posición que ocupa la clave a extraer (E).
{
  Nodo.nK--;
  for (i = Pos; i <= Nodo.nK; i++)
     { Nodo.K[i] = Nodo.K[i + 1];
       Nodo.A[i] = Nodo.A[i + 1];
     }
  return;
}
```

5. ALGORITMO DE DESPLAZAMIENTO DE CLAVES

```
Procedimiento DesplazarClaves(Nodo, Pos, Num, Sentido)
Nodo: sobre el que se realiza el desplazamiento (E/S).
Pos: primera posición desde la que se realiza el desplazamiento (E).
Num: número de posiciones a desplazar (E).
Sentido: indica el sentido IZQUIERDA/DERECHA del desplazamiento (E).
{
  if (Sentido == DERECHA)
     { for (i = Nodo.nK; i >= Pos; i--)
          { Nodo.K[i + Num] = Nodo.K[i];
            Nodo.A[i + Num] = Nodo.A[i];
          }
     }
  else
     { for (i = Pos; i <= Nodo.nK; i++)
          { Nodo.K[i - Num] = Nodo.K[i];
            Nodo.A[i - Num] = Nodo.A[i];
          }
     }
  return;
}
```

6. ALGORITMO DE MOVIMIENTO DE CLAVES ENTRE NODOS

```
Procedimiento MoverClaves(NodoFuente, NodoDestino, Pos, Num, Sentido)
NodoFuente: desde el que se mueven las claves (E/S).
NodoDestino: al que le llegan las claves (E/S).
Pos: posición inicial de las claves que se moveran (E).
Num: número de claves a mover (E).
Sentido: indica el sentido IZQUIERDA/DERECHA del movimiento (E).
{
  for (i = 1; i <= Num; i++)
     { if (Sentido == DERECHA)
          { NodoDestino.K[i] = NodoFuente.K[Pos + i - 1];
            NodoDestino.A[i] = NodoFuente.A[Pos + i - 1];
          }
       else
          { NodoDestino.K[NodoDestino.nK + i] = NodoFuente.K[Pos + i - 1];
            NodoDestino.A[NodoDestino.nK + i] = NodoFuente.A[Pos + i - 1];
          }
     }
  return;
}
```

Algoritmos de inserción

1. PARTICIÓN-1/2

- Situación preliminar

$$\mathbf{nP = m}$$

$$\mathbf{F[nF;\ldots\ldots,AF(jF),KF(jF+1),\ldots\ldots]}$$

$$\mathbf{P[nP;AP(0),KP(1),\ldots\ldots,KP(nP),AP(nP)]}$$

- Situación posterior

$$\mathbf{F\left[nF+1;\ldots\ldots,AF(jF),KP\left(\left\lceil\frac{m}{2}\right\rceil\right),AP',KF(jF+1),\ldots\ldots\right]}$$

$$\mathbf{P\left[\left\lceil\frac{m}{2}\right\rceil-1;AP(0),KP(1),\ldots\ldots,KP\left(\left\lceil\frac{m}{2}\right\rceil-1\right),AP\left(\left\lceil\frac{m}{2}\right\rceil-1\right)\right]}$$

$$\mathbf{P'\left[m-\left\lceil\frac{m}{2}\right\rceil;AP\left(\left\lceil\frac{m}{2}\right\rceil\right),KP\left(\left\lceil\frac{m}{2}\right\rceil+1\right),AP\left(\left\lceil\frac{m}{2}\right\rceil+1\right),\ldots\ldots,KP(nP),AP(nP)\right]}$$

- Procedimiento Partición-1/2

```
Procedimiento Particion-1/2(NodoP, Clave, Puntero)
NodoP: nodo donde se realizó la inserción (E).
Clave: valor de clave a insertar en el nivel superior (S).
Puntero: dirección del nuevo nodo creado (S).
{
  Clave = NodoP.K[⌈m/2.0⌉];

  NodoP'.Direccion = Puntero = CrearNuevoNodo();
  NodoP'.A[0] = NodoP.A[⌈m/2.0⌉];
  MoverClaves(NodoP, NodoP', ⌈m/2.0⌉ + 1, m - ⌈m/2.0⌉, DERECHA);
  NodoP'.nK = m - ⌈m/2.0⌉;

  NodoP.nK = ⌈m/2.0⌉ - 1;

  EscribirNodo(NodoP);
  EscribirNodo(NodoP');
  return;
}
```

2. ALGORITMO DE INSERCIÓN EN UN ÁRBOL-B (PARTICIÓN-1/2)

```
Funcion InsertarAB1(R, X)
R: raíz del árbol (E/S).
X: clave a insertar (E).
{
  if (BuscarAMC(R, X, Pila, Top))
     return FALSE;

  Clave = X;
  Puntero = Nulo;
  while (Top >= 0)
     { NodoP = Pila[Top].Nodo;
       jP = Pila[Top].Posicion;
       Top--;
       InsertarClaveYPuntero(NodoP, jP + 1, Clave, Puntero);
       if (NodoP.nK <= m - 1)
          { EscribirNodo(NodoP);
            return TRUE;
          }
       Particion-1/2(NodoP, Clave, Puntero);
     }

  NodoR.Direccion = CrearNuevoNodo();
  NodoR.A[0] = R;
  NodoR.K[1] = Clave;
  NodoR.A[1] = Puntero;
  NodoR.nK = 1;

  R = NodoR.Direccion;

  EscribirNodo(NodoR);
  return TRUE;
}
```

3. DROTACIÓN-I/1

- Situación preliminar

$$\mathbf{nP = m}$$

$$\mathbf{nD < m - 1}$$

$$\mathbf{F[nF;\ldots..,AF(jF),KF(jF+1),AF(jF+1),\ldots..]}$$

$$\mathbf{P[nP;AP(0),KP(1),\ldots..,KP(nP),AP(nP)]}$$

$$\mathbf{D[nD;AD(0),KD(1),\ldots..,KD(nD),AD(nD)]}$$

- Situación posterior

$$\mathbf{F[nF;\ldots..,AF(jF),KP(nP),AF(jF+1),\ldots..]}$$

$$\mathbf{P[nP-1;AP(0),KP(1),\ldots..,KP(nP-1),AP(nP-1)]}$$

$$\mathbf{D[nD+1;AP(nP),KF(jF+1),AD(0),KD(1),\ldots..,KD(nD),AD(nD)]}$$

- Procedimiento DRotacion-I/1

```
Procedimiento DRotacion-I/1(NodoP, NodoF, NodoD, jF)
NodoP: nodo sobrecargado (E).
NodoF: padre de NodoP (E).
NodoD: hermano inmediato derecho de NodoP (E).
jF: posición dentro de NodoF del puntero a NodoP (E).
{
  DesplazarClaves(NodoD, 1, 1, DERECHA);
  NodoD.A[1] = NodoD.A[0];
  NodoD.A[0] = NodoP.A[NodoP.nK];
  NodoD.K[1] = NodoF.K[jF + 1];
  NodoD.nK++;

  NodoF.K[jF + 1] = NodoP.K[NodoP.nK];

  NodoP.nK--;

  EscribirNodo(NodoP);
  EscribirNodo(NodoF);
  EscribirNodo(NodoD);
  return;
}
```

4. IROTACIÓN-I/1

- Situación preliminar

$$\mathbf{nP = m}$$

$$\mathbf{nI < m - 1}$$

$$\mathbf{F[nF;\ldots..,AF(jF-1),KF(jF),AF(jF),\ldots..]}$$

$$\mathbf{I[nI;AI(0),KI(1),\ldots..,KI(nI),AI(nI)]}$$

$$\mathbf{P[nP;AP(0),KP(1),\ldots..,KP(nP),AP(nP)]}$$

- Situación posterior

$$\mathbf{F[nF;\ldots..,AF(jF-1),KP(1),AF(jF),\ldots..]}$$

$$\mathbf{I[nI+1;AI(0),KI(1),\ldots..,KI(nI),AI(nI),KF(jF),AP(0)]}$$

$$\mathbf{P[nP-1;AP(1),KP(2),\ldots..,KP(nP),AP(nP)]}$$

- Procedimiento IRotacion-I/1

```
Procedimiento IRotacion-I/1(NodoI, NodoF, NodoP, jF)
NodoI: hermano inmediato izquierdo de NodoP (E).
NodoF: padre de NodoP (E).
NodoP: nodo sobrecargado (E).
jF: posición dentro de NodoF del puntero a NodoP (E).
{
  NodoI.nK++;
  NodoI.K[NodoI.nK] = NodoF.K[jF];
  NodoI.A[NodoI.nK] = NodoP.A[0];

  NodoF.K[jF] = NodoP.K[1];

  NodoP.A[0] = NodoP.A[1];
  DesplazarClaves(NodoP, 2, 1, IZQUIERDA);
  NodoP.nK--;

  EscribirNodo(NodoI);
  EscribirNodo(NodoF);
  EscribirNodo(NodoP);
  return;
}
```

5. DROTACIÓN-I/N

- Situación preliminar

$$nP = m$$

$$nD < m - 1$$

$$F[nF;\ldots\ldots, AF(jF), KF(jF+1), AF(jF+1),\ldots\ldots]$$

$$P[nP; AP(0), KP(1),\ldots\ldots, KP(nP), AP(nP)]$$

$$D[nD; AD(0), KD(1),\ldots\ldots, KD(nD), AD(nD)]$$

- Situación posterior

$$F[nF;\ldots\ldots, AF(jF), KP(\lfloor \tfrac{nP+nD}{2} \rfloor + 1), AF(jF+1),\ldots\ldots]$$

$$P[\lfloor \tfrac{nP+nD}{2} \rfloor; AP(0), KP(1),\ldots\ldots, KP(\lfloor \tfrac{nP+nD}{2} \rfloor), AP(\lfloor \tfrac{nP+nD}{2} \rfloor)]$$

$$D[\lceil \tfrac{nP+nD}{2} \rceil; AP(\lfloor \tfrac{nP+nD}{2} \rfloor + 1), KP(\lfloor \tfrac{nP+nD}{2} \rfloor + 2),\ldots\ldots, AP(nP), KF(jF+1), AD(0),\ldots\ldots, AD(nD)]$$

- Procedimiento DRotacion-I/N

```
Procedimiento DRotacion-I/N(NodoP, NodoF, NodoD, jF)
NodoP: nodo sobrecargado (E).
NodoF: padre de NodoP (E).
NodoD: hermano inmediato derecho de NodoP (E).
jF: posición dentro de NodoF del puntero a NodoP (E).
{
  nKP = ⌊(NodoP.nK + NodoD.nK)/2.0⌋;
  nKD = ⌈(NodoP.nK + NodoD.nK)/2.0⌉;
  nKDif = nKD – NodoD.nK;

  DesplazarClaves(NodoD, 1, nKDif, DERECHA);
  NodoD.K[nKDif] = NodoF.K[jF + 1];
  NodoD.A[nKDif] = NodoD.A[0];
  NodoD.A[0] = NodoP.A[nKP + 1];
  MoverClaves(NodoP, NodoD, nKP + 2, nKDif - 1, DERECHA);
  NodoD.nK = nKD;

  NodoF.K[jF + 1] = NodoP.K[nKP + 1];

  NodoP.nK = nKP;

  EscribirNodo(NodoP);
  EscribirNodo(NodoF);
  EscribirNodo(NodoD);
  return;
}
```

6. IROTACIÓN-I/N

- Situación preliminar

$$\mathbf{nP = m}$$

$$\mathbf{nI < m - 1}$$

$$\mathbf{F[nF;\ldots\ldots, AF(jF-1), KF(jF), AF(jF), \ldots\ldots]}$$

$$\mathbf{I[nI; AI(0), KI(1), \ldots\ldots, KI(nI), AI(nI)]}$$

$$\mathbf{P[nP; AP(0), KP(1), \ldots\ldots, KP(nP), AP(nP)]}$$

- Situación posterior

$$\mathbf{F[nF;\ldots\ldots, AF(jF-1), KP\left(m - \left\lfloor \frac{nP+nI}{2} \right\rfloor\right), AF(jF), \ldots\ldots]}$$

$$\mathbf{I\left[\left\lceil \frac{nP+nI}{2} \right\rceil; AI(0), \ldots\ldots, AI(nI), KF(jF), AP(0), \ldots\ldots, KP\left(m - \left\lfloor \frac{nP+nI}{2} \right\rfloor - 1\right), AP\left(m - \left\lfloor \frac{nP+nI}{2} \right\rfloor - 1\right)\right]}$$

$$\mathbf{P\left[\left\lfloor \frac{nP+nI}{2} \right\rfloor; AP\left(m - \left\lfloor \frac{nP+nI}{2} \right\rfloor\right), KP\left(m - \left\lfloor \frac{nP+nI}{2} \right\rfloor + 1\right), \ldots\ldots, KP(nP), AP(nP)\right]}$$

- Procedimiento IRotacion-I/N

```
Procedimiento IRotacion-I/N(NodoI, NodoF, NodoP, jF)
NodoI: hermano inmediato izquierdo de NodoP (E).
NodoF: padre de NodoP (E).
NodoD: nodo sobrecargado (E).
jF: posición dentro de NodoF del puntero a NodoP (E).
{
  nKP = ⌊(NodoP.nK + NodoI.nK)/2.0⌋;
  nKI = ⌈(NodoP.nK + NodoI.nK)/2.0⌉;
  nKDif = nKI – NodoI.nK;
  NodoI.nK++;
  NodoI.K[NodoI.nK] = NodoF.K[jF];
  NodoI.A[NodoI.nK] = NodoP.A[0];
  MoverClaves(NodoP, NodoI, 1, nKDif – 1, IZQUIERDA);
  NodoI.nK = nKI;

  NodoF.K[jF] = NodoP.K[nKDif];

  NodoP.A[0] = NodoP.A[nKDif];
  DesplazarClaves(NodoP, nKDif + 1, nKDif, IZQUIERDA);
  NodoP.nK = nKP;

  EscribirNodo(NodoI);
  EscribirNodo(NodoF);
  EscribirNodo(NodoP);
  return;
}
```

7. ALGORITMO DE INSERCIÓN EN UN ÁRBOL-B (ROTACION-I/N)

```
Funcion InsertarAB2(R, X)
R: raíz del árbol (E/S).
X: clave a insertar (E).
{
  Clave = X;
  Puntero = Nulo;
  if (R != Nulo)
     { if (BuscarAMC(R, X, Pila, Top))
          return FALSE;
       NodoP = Pila[Top].Nodo;
       jP = Pila[Top].Posicion;
       Top--;
       InsertarClaveYPuntero(NodoP, jP + 1, Clave, Puntero);
       if (NodoP.nK <= m - 1)
          { EscribirNodo(NodoP); return TRUE;
          }
       while (Top >= 0)
          { NodoF = Pila[Top].Nodo;
            jF = Pila[Top].Posicion;
            Top--;
            if (jF < NodoF.nK)
               { LeerNodo(NodoF.A[jF + 1], NodoD);
                 if (NodoD.nK < m - 1)
                    { DRotacion-I/N(NodoP, NodoF, NodoD, jF);
                      return TRUE;
                    }
               }
            if (jF > 0)
               { LeerNodo(NodoF.A[jF - 1], NodoI);
                 if (NodoI.nK < m - 1)
                    { IRotacion-I/N(NodoI, NodoF, NodoP, jF);
                      return TRUE;
                    }
               }
            Particion-1/2(NodoP, Clave, Puntero);
            NodoP = NodoF;
            jP = jF;
            InsertarClaveYPuntero(NodoP, jP + 1, Clave, Puntero);
            if (NodoP.nK <= m - 1)
               { EscribirNodo(NodoP); return TRUE;
               }
          }
       Particion-1/2(NodoP, Clave, Puntero);
     }
  NodoR.Direccion = CrearNuevoNodo();
  NodoR.A[0] = R;
  NodoR.K[1] = Clave;
  NodoR.A[1] = Puntero;
  NodoR.nK = 1;
  R = NodoR.Direccion;
  EscribirNodo(NodoR);
  return TRUE;
}
```

Algoritmos de extracción

1. DROTACIÓN-E/1

- Situación preliminar

$$\mathbf{nP} = \left\lceil \frac{\mathbf{m}}{2} \right\rceil - \mathbf{2}$$

$$\mathbf{nD} > \left\lceil \frac{\mathbf{m}}{2} \right\rceil - \mathbf{1}$$

$$\mathbf{F}\left[\mathbf{nF};\ldots\ldots,\mathbf{AF}(\mathbf{jF}),\mathbf{KF}(\mathbf{jF}+\mathbf{1}),\mathbf{AF}(\mathbf{jF}+\mathbf{1}),\ldots\ldots\right]$$

$$\mathbf{P}\left[\mathbf{nP};\mathbf{AP}(\mathbf{0}),\mathbf{KP}(\mathbf{1}),\ldots\ldots,\mathbf{KP}(\mathbf{nP}),\mathbf{AP}(\mathbf{nP})\right]$$

$$\mathbf{D}\left[\mathbf{nD};\mathbf{AD}(\mathbf{0}),\mathbf{KD}(\mathbf{1}),\ldots\ldots,\mathbf{KD}(\mathbf{nD}),\mathbf{AD}(\mathbf{nD})\right]$$

- Situación posterior

$$\mathbf{F}\left[\mathbf{nF};\ldots\ldots,\mathbf{AF}(\mathbf{jF}),\mathbf{KD}(\mathbf{1}),\mathbf{AF}(\mathbf{jF}+\mathbf{1}),\ldots\ldots\right]$$

$$\mathbf{P}\left[\mathbf{nP}+\mathbf{1};\mathbf{AP}(\mathbf{0}),\mathbf{KP}(\mathbf{1}),\ldots\ldots,\mathbf{KP}(\mathbf{nP}),\mathbf{AP}(\mathbf{nP}),\mathbf{KF}(\mathbf{jF}+\mathbf{1}),\mathbf{AD}(\mathbf{0})\right]$$

$$\mathbf{D}\left[\mathbf{nD}-\mathbf{1};\mathbf{AD}(\mathbf{1}),\mathbf{KD}(\mathbf{2}),\ldots\ldots,\mathbf{KD}(\mathbf{nD}),\mathbf{AD}(\mathbf{nD})\right]$$

- Procedimiento DRotacion-E/1

```
Procedimiento DRotacion-E/1(NodoP, NodoF, NodoD, jF)
NodoP: nodo bajo mínimos (E).
NodoF: padre de NodoP (E).
NodoD: hermano inmediato derecho de NodoP (E).
jF: posición dentro de NodoF del puntero a NodoP (E).
{
  NodoP.nK++;
  NodoP.K[NodoP.nK] = NodoF.K[jF + 1];
  NodoP.A[NodoP.nK] = NodoD.A[0];

  NodoF.K[jF + 1] = NodoD.K[1];

  NodoD.A[0] = NodoD.A[1];
  DesplazarClaves(NodoD, 2, 1, IZQUIERDA);
  NodoD.nK--;

  EscribirNodo(NodoP);
  EscribirNodo(NodoF);
  EscribirNodo(NodoD);
  return;
}
```

2. IROTACIÓN-E/1

- Situación preliminar

$$\mathbf{nP} = \left\lceil \frac{\mathbf{m}}{2} \right\rceil - \mathbf{2}$$

$$\mathbf{nI} > \left\lceil \frac{\mathbf{m}}{2} \right\rceil - \mathbf{1}$$

$$\mathbf{F}\left[\mathbf{nF};\ldots\ldots,\mathbf{AF}(\mathbf{jF}-\mathbf{1}),\mathbf{KF}(\mathbf{jF}),\mathbf{AF}(\mathbf{jF}),\ldots\ldots\right]$$

$$\mathbf{I}\left[\mathbf{nI};\mathbf{AI}(\mathbf{0}),\mathbf{KI}(\mathbf{1}),\ldots\ldots,\mathbf{KI}(\mathbf{nI}),\mathbf{AI}(\mathbf{nI})\right]$$

$$\mathbf{P}\left[\mathbf{nP};\mathbf{AP}(\mathbf{0}),\mathbf{KP}(\mathbf{1}),\ldots\ldots,\mathbf{KP}(\mathbf{nP}),\mathbf{AP}(\mathbf{nP})\right]$$

- Situación posterior

$$\mathbf{F}\left[\mathbf{nF};\ldots\ldots,\mathbf{AF}(\mathbf{jF}-\mathbf{1}),\mathbf{KI}(\mathbf{nI}),\mathbf{AF}(\mathbf{jF}),\ldots\ldots\right]$$

$$\mathbf{I}\left[\mathbf{nI}-\mathbf{1};\mathbf{AI}(\mathbf{0}),\mathbf{KI}(\mathbf{1}),\ldots\ldots,\mathbf{KI}(\mathbf{nI}-\mathbf{1}),\mathbf{AI}(\mathbf{nI}-\mathbf{1})\right]$$

$$\mathbf{P}\left[\mathbf{nP}+\mathbf{1};\mathbf{AI}(\mathbf{nI}),\mathbf{KF}(\mathbf{jF}),\mathbf{AP}(\mathbf{0}),\mathbf{KP}(\mathbf{1}),\ldots\ldots,\mathbf{KP}(\mathbf{nP}),\mathbf{AP}(\mathbf{nP})\right]$$

- Procedimiento IRotacion-E/1

```
Procedimiento IRotacion-E/1(NodoI, NodoF, NodoP, jF)
NodoI: hermano inmediato derecho de NodoP (E).
NodoF: padre de NodoP (E).
NodoP: nodo bajo mínimos (E).
jF: posición dentro de NodoF del puntero a NodoP (E).
{
  DesplazarClaves(NodoP, 1, 1, DERECHA);
  NodoP.A[1] = NodoP.A[0];
  NodoP.A[0] = NodoI.A[NodoI.nK];
  NodoP.K[1] = NodoF.K[jF];
  NodoP.nK++;

  NodoF.K[jF] = NodoI.K[NodoI.nK];

  NodoI.nK--;

  EscribirNodo(NodoI);
  EscribirNodo(NodoF);
  EscribirNodo(NodoP);
  return;
}
```

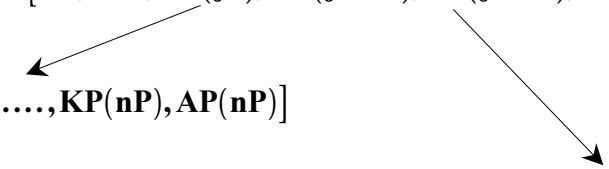
3. DRECOMBINACIÓN-2/1

- Situación preliminar

$$\mathbf{nP} = \left\lceil \frac{\mathbf{m}}{2} \right\rceil - \mathbf{2}$$

$$\mathbf{nD} = \left\lceil \frac{\mathbf{m}}{2} \right\rceil - \mathbf{1}$$

$$\mathbf{F}\left[\mathbf{nF};\ldots\ldots,\mathbf{AF}(\mathbf{jF}),\mathbf{KF}(\mathbf{jF}+\mathbf{1}),\mathbf{AF}(\mathbf{jF}+\mathbf{1}),\ldots\ldots\right]$$



$$\mathbf{P}\left[\mathbf{nP};\mathbf{AP}(\mathbf{0}),\mathbf{KP}(\mathbf{1}),\ldots\ldots,\mathbf{KP}(\mathbf{nP}),\mathbf{AP}(\mathbf{nP})\right]$$

$$\mathbf{D}\left[\mathbf{nD};\mathbf{AD}(\mathbf{0}),\mathbf{KD}(\mathbf{1}),\ldots\ldots,\mathbf{KD}(\mathbf{nD}),\mathbf{AD}(\mathbf{nD})\right]$$

- Situación posterior

$$\mathbf{F}\left[\mathbf{nF}-\mathbf{1};\ldots\ldots,\mathbf{AF}(\mathbf{jF}),\mathbf{KF}(\mathbf{jF}+\mathbf{2}),\mathbf{AF}(\mathbf{jF}+\mathbf{2}),\ldots\ldots\right]$$

$$\mathbf{P}\left[\mathbf{nP}+\mathbf{nD}+\mathbf{1};\mathbf{AP}(\mathbf{0}),\mathbf{KP}(\mathbf{1}),\ldots\ldots,\mathbf{KP}(\mathbf{nP}),\mathbf{AP}(\mathbf{nP}),\mathbf{KF}(\mathbf{jF}+\mathbf{1}),\mathbf{AD}(\mathbf{0}),\mathbf{KD}(\mathbf{1}),\ldots\ldots,\mathbf{KD}(\mathbf{nD}),\mathbf{AD}(\mathbf{nD})\right]$$

- Procedimiento DRecombinacion-2/1

```
Procedimiento DRecombinacion-2/1(NodoP, NodoF, NodoD, jF)
NodoP: nodo bajo mínimos (E).
NodoF: padre de NodoP (E/S).
NodoD: hermano inmediato derecho de NodoP (E).
jF: posición dentro de NodoF del puntero a NodoP (E).
{
  NodoP.nK++;
  NodoP.K[NodoP.nK] = NodoF.K[jF + 1];
  NodoP.A[NodoP.nK] = NodoD.A[0];
  MoverClaves(NodoD, NodoP, 1, NodoD.nK, IZQUIERDA);

  NodoP.nK += NodoD.nK;

  ExtraerClave(NodoF, jF + 1);
  ExtraerNodo(NodoD);
  EscribirNodo(NodoP);
  EscribirNodo(NodoF);
  return;
}
```

4. IRECOMBINACIÓN-2/1

- Situación preliminar

$$\mathbf{nP} = \left\lceil \frac{\mathbf{m}}{2} \right\rceil - \mathbf{2}$$

$$\mathbf{nI} = \left\lceil \frac{\mathbf{m}}{2} \right\rceil - \mathbf{1}$$

$$\mathbf{F}\left[\mathbf{nF}; \ldots\ldots, \mathbf{AF}(\mathbf{jF}-\mathbf{1}), \mathbf{KF}(\mathbf{jF}), \mathbf{AF}(\mathbf{jF}), \ldots\ldots\right]$$

$$\mathbf{I}\left[\mathbf{nI}; \mathbf{AI}(\mathbf{0}), \mathbf{KI}(\mathbf{1}), \ldots\ldots, \mathbf{KI}(\mathbf{nI}), \mathbf{AI}(\mathbf{nI})\right]$$

$$\mathbf{P}\left[\mathbf{nP}; \mathbf{AP}(\mathbf{0}), \mathbf{KP}(\mathbf{1}), \ldots\ldots, \mathbf{KP}(\mathbf{nP}), \mathbf{AP}(\mathbf{nP})\right]$$

- Situación posterior

$$\mathbf{F}\left[\mathbf{nF}-\mathbf{1}; \ldots\ldots, \mathbf{AF}(\mathbf{jF}-\mathbf{1}), \mathbf{KF}(\mathbf{jF}+\mathbf{1}), \mathbf{AF}(\mathbf{jF}+\mathbf{1}), \ldots\ldots\right]$$

$$\mathbf{I}\left[\mathbf{nI}+\mathbf{nP}+\mathbf{1}; \mathbf{AI}(\mathbf{0}), \mathbf{KI}(\mathbf{1}), \ldots\ldots, \mathbf{KI}(\mathbf{nI}), \mathbf{AI}(\mathbf{nI}), \mathbf{KF}(\mathbf{jF}), \mathbf{AP}(\mathbf{0}), \mathbf{KP}(\mathbf{1}), \ldots\ldots, \mathbf{KP}(\mathbf{nP}), \mathbf{AP}(\mathbf{nP})\right]$$

- Procedimiento IRecombiacion-2/1

```
Procedimiento IRecombinacion-2/1(NodoI, NodoF, NodoP, jF)
NodoI: hermano inmediato izquierdo de NodoP (E).
NodoF: padre de NodoP (E/S).
NodoP: nodo bajo mínimos (E).
jF: posición dentro de NodoF del puntero a NodoP (E).
{
  NodoI.nK++;
  NodoI.K[NodoI.nK] = NodoF.K[jF];
  NodoI.A[NodoI.nK] = NodoP.A[0];
  MoverClaves(NodoP, NodoI, 1, NodoP.nK, IZQUIERDA);

  NodoI.nK += NodoP.nK;

  ExtraerClave(NodoF, jF);
  ExtraerNodo(NodoP);
  EscribirNodo(NodoI);
  EscribirNodo(NodoF);
  return;
}
```

5. ALGORITMO DE EXTRACCIÓN EN ÁRBOL-B (ROTACIÓN-E/1, RECOMBINACIÓN-2/1)

```
Funcion ExtraerAB1(R, X)
R: raíz del árbol (E/S).
X: clave a extraer (E).
{
  if (!BuscarAMC(R, X, Pila, Top))
     return FALSE;
  NodoP = Pila[Top].Nodo;
  Pos = Pila[Top].Posicion;
  if (NodoP.A[Pos] != Nulo)
     { TopP = Top;
       LeerNodo(NodoP.A[Pos], NodoQ);
       Top++;
       Pila[Top].Nodo = NodoQ;;
       Pila[Top].Posicion = 0;
       while (NodoQ.A[0] != Nulo)
          { LeerNodo(NodoQ.A[0], NodoQ);
            Top++;
            Pila[Top].Nodo = NodoQ;
            Pila[Top].Posicion = 0;
          }
       Pila[TopP].Nodo.K[Pos] = NodoP.K[Pos] = NodoQ.K[1];
       EscribirNodo(NodoP);
       Pos = 1;
       NodoP = NodoQ;
     }
  ExtraerClave(NodoP, Pos);
  while ((NodoP.nK < ⌈m/2.0⌉ - 1) && (NodoP.Direccion != R))
     { Top--;
       NodoF = Pila[Top].Nodo;
       jF = Pila[Top].Posicion;
       if (jF < NodoF.nK)
          { LeerNodo(NodoF.A[jF + 1], NodoD);
            if (NodoD.nK > ⌈m/2.0⌉ - 1)
               { DRotacion-E/1(NodoP, NodoF, NodoD, jF);
                 return TRUE;
               }
          }
       if (jF > 0)
          { LeerNodo(NodoF.A[jF - 1], NodoI);
            if (NodoI.nK > ⌈m/2.0⌉ - 1)
               { IRotacion-E/1(NodoI, NodoF, NodoP, jF);
                 return TRUE;
               }
          }
       if (jF == 0) DRecombinacion-2/1(NodoP, NodoF, NodoD, jF);
       else         IRecombinacion-2/1(NodoI, NodoF, NodoP, jF);
       NodoP = NodoF;
     }
  if (NodoP.nK != 0)
     EscribirNodo(NodoP);
  else
     { R = NodoP.A[0];
       ExtraerNodo(NodoP);
     }
  return TRUE;
}
```

6. DROTACIÓN-E/N

- Situación preliminar

$$\mathbf{nP} = \left\lceil \frac{\mathbf{m}}{2} \right\rceil - \mathbf{2}$$

$$\mathbf{nD} > \left\lceil \frac{\mathbf{m}}{2} \right\rceil - \mathbf{1}$$

$$\mathbf{F}\left[\mathbf{nF};\ldots\ldots,\mathbf{AF}(\mathbf{jF}),\mathbf{KF}(\mathbf{jF}+\mathbf{1}),\mathbf{AF}(\mathbf{jF}+\mathbf{1}),\ldots\ldots\right]$$

$$\mathbf{P}\left[\mathbf{nP};\mathbf{AP}(\mathbf{0}),\mathbf{KP}(\mathbf{1}),\ldots\ldots,\mathbf{KP}(\mathbf{nP}),\mathbf{AP}(\mathbf{nP})\right]$$

$$\mathbf{D}\left[\mathbf{nD};\mathbf{AD}(\mathbf{0}),\mathbf{KD}(\mathbf{1}),\ldots\ldots,\mathbf{KD}(\mathbf{nD}),\mathbf{AD}(\mathbf{nD})\right]$$

- Situación posterior

$$\mathbf{F}\left[\mathbf{nF};\ldots\ldots,\mathbf{AF}(\mathbf{jF}),\mathbf{KD}\left(\mathbf{nD}-\left\lfloor\frac{\mathbf{nP}+\mathbf{nD}}{\mathbf{2}}\right\rfloor\right),\mathbf{AF}(\mathbf{jF}+\mathbf{1}),\ldots\ldots\right]$$

$$\mathbf{P}\left[\left\lceil\frac{\mathbf{nP}+\mathbf{nD}}{\mathbf{2}}\right\rceil;\mathbf{AP}(\mathbf{0}),\ldots\ldots,\mathbf{AP}(\mathbf{nP}),\mathbf{KF}(\mathbf{jF}+\mathbf{1}),\mathbf{AD}(\mathbf{0}),\ldots\ldots,\mathbf{AD}\left(\mathbf{nD}-\left\lfloor\frac{\mathbf{nP}+\mathbf{nD}}{\mathbf{2}}\right\rfloor-\mathbf{1}\right)\right]$$

$$\mathbf{D}\left[\left\lfloor\frac{\mathbf{nP}+\mathbf{nD}}{\mathbf{2}}\right\rfloor;\mathbf{AD}\left(\mathbf{nD}-\left\lfloor\frac{\mathbf{nP}+\mathbf{nD}}{\mathbf{2}}\right\rfloor\right),\mathbf{KD}\left(\mathbf{nD}-\left\lfloor\frac{\mathbf{nP}+\mathbf{nD}}{\mathbf{2}}\right\rfloor+\mathbf{1}\right),\ldots\ldots,\mathbf{KD}(\mathbf{nD}),\mathbf{AD}(\mathbf{nD})\right]$$

- Procedimiento DRotacion-E/N

```
Procedimiento DRotacion-E/N(NodoP, NodoF, NodoD, jF)
NodoP: nodo bajo mínimos (E).
NodoF: padre de NodoP (E).
NodoD: hermano inmediato derecho de NodoP (E).
jF: posición dentro de NodoF del puntero a NodoP (E).
{
  nKP = ⌈(NodoP.nK + NodoD.nK)/2.0⌉;
  nKD = ⌊(NodoP.nK + NodoD.nK)/2.0⌋;
  nKDif = nKP – NodoP.nK;
  NodoP.nK++;
  NodoP.K[NodoP.nK] = NodoF.K[jF + 1];
  NodoP.A[NodoP.nK] = NodoD.A[0];
  MoverClaves(NodoD, NodoP, 1, nKDif – 1, IZQUIERDA);
  NodoP.nK = nKP;

  NodoF.K[jF + 1] = NodoD.K[nKDif];

  NodoD.A[0] = NodoD.A[nKDif];
  DesplazarClaves(NodoD, nKDif + 1, nKDif, IZQUIERDA);
  NodoD.nK = nKD;
  EscribirNodo(NodoP);
  EscribirNodo(NodoF);
  EscribirNodo(NodoD);
  return;
}
```

7. IROTACIÓN-E/N

- Situación preliminar

$$\mathbf{nP} = \left\lceil \frac{\mathbf{m}}{2} \right\rceil - \mathbf{2}$$

$$\mathbf{nI} > \left\lceil \frac{\mathbf{m}}{2} \right\rceil - \mathbf{1}$$

$$\mathbf{F}\left[\mathbf{nF};\ldots\ldots,\mathbf{AF}(\mathbf{jF}-\mathbf{1}),\mathbf{KF}(\mathbf{jF}),\mathbf{AF}(\mathbf{jF}),\ldots\ldots\right]$$

$$\mathbf{I}\left[\mathbf{nI};\mathbf{AI}(\mathbf{0}),\mathbf{KI}(\mathbf{1}),\ldots\ldots,\mathbf{KI}(\mathbf{nI}),\mathbf{AI}(\mathbf{nI})\right]$$

$$\mathbf{P}\left[\mathbf{nP};\mathbf{AP}(\mathbf{0}),\mathbf{KP}(\mathbf{1}),\ldots\ldots,\mathbf{KP}(\mathbf{nP}),\mathbf{AP}(\mathbf{nP})\right]$$

- Situación posterior

$$\mathbf{F}\left[\mathbf{nF};\ldots\ldots,\mathbf{AF}(\mathbf{jF}-\mathbf{1}),\mathbf{KI}\left(\left\lfloor\frac{\mathbf{nP}+\mathbf{nI}}{\mathbf{2}}\right\rfloor+\mathbf{1}\right),\mathbf{AF}(\mathbf{jF}),\ldots\ldots\right]$$

$$\mathbf{I}\left[\left\lfloor\frac{\mathbf{nP}+\mathbf{nI}}{\mathbf{2}}\right\rfloor;\mathbf{AI}(\mathbf{0}),\mathbf{KI}(\mathbf{1}),\ldots\ldots,\mathbf{KI}\left(\left\lfloor\frac{\mathbf{nP}+\mathbf{nI}}{\mathbf{2}}\right\rfloor\right),\mathbf{AI}\left(\left\lfloor\frac{\mathbf{nP}+\mathbf{nI}}{\mathbf{2}}\right\rfloor\right)\right]$$

$$\mathbf{P}\left[\left\lceil\frac{\mathbf{nP}+\mathbf{nI}}{\mathbf{2}}\right\rceil;\mathbf{AI}\left(\left\lfloor\frac{\mathbf{nP}+\mathbf{nI}}{\mathbf{2}}\right\rfloor+\mathbf{1}\right),\mathbf{KI}\left(\left\lfloor\frac{\mathbf{nP}+\mathbf{nI}}{\mathbf{2}}\right\rfloor+\mathbf{2}\right),\ldots\ldots,\mathbf{AI}(\mathbf{nI}),\mathbf{KF}(\mathbf{jF}),\mathbf{AP}(\mathbf{0}),\ldots\ldots,\mathbf{AP}(\mathbf{nP})\right]$$

- Procedimiento IRotacion-E/N

```
Procedimiento IRotacion-E/N(NodoI, NodoF, NodoP, jF)
NodoI: hermano inmediato izquierdo de NodoP (E).
NodoF: padre de NodoP (E).
NodoP: nodo bajo mínimos (E).
jF: posición dentro de NodoF del puntero a NodoP (E).
{
  nKP = ⌈(NodoP.nK + NodoI.nK)/2.0⌉;
  nKI = ⌊(NodoP.nK + NodoI.nK)/2.0⌋;
  nKDif = nKP – NodoP.nK;

  DesplazarClaves(NodoP, 1, nKDif, DERECHA);
  NodoP.A[nKDif] = NodoP.A[0];
  NodoP.K[nKDif] = NodoF.K[jF];
  NodoP.A[0] = NodoI.A[nKI + 1];
  MoverClaves(NodoI, NodoP, nKI + 2, nKDif – 1, DERECHA);
  NodoP.nK = nKP;

  NodoF.K[jF] = NodoI.K[nKI + 1];

  NodoI.nK = nKI;

  EscribirNodo(NodoI);
  EscribirNodo(NodoF);
  EscribirNodo(NodoP);
  return;
}
```

8. RECOMBINACIÓN-3/2

- Situación preliminar

$$\mathbf{nP} = \left\lceil \frac{m}{2} \right\rceil - 2$$

$$\mathbf{nI} = \mathbf{nD} = \left\lceil \frac{m}{2} \right\rceil - 1$$

$$\mathbf{F}\left[\mathbf{nF};\ldots\ldots,\mathbf{AF}(\mathbf{jF}-1),\mathbf{KF}(\mathbf{jF}),\mathbf{AF}(\mathbf{jF}),\mathbf{KF}(\mathbf{jF}+1),\mathbf{AF}(\mathbf{jF}+1),\ldots\ldots\right]$$

$$\mathbf{I}\left[\mathbf{nI};\mathbf{AI}(0),\mathbf{KI}(1),\ldots\ldots,\mathbf{KI}(\mathbf{nI}),\mathbf{AI}(\mathbf{nI})\right]$$

$$\mathbf{P}\left[\mathbf{nP};\mathbf{AP}(0),\mathbf{KP}(1),\ldots\ldots,\mathbf{KP}(\mathbf{nP}),\mathbf{AP}(\mathbf{nP})\right]$$

$$\mathbf{D}\left[\mathbf{nD};\mathbf{AD}(0),\mathbf{KD}(1),\ldots\ldots,\mathbf{KD}(\mathbf{nD}),\mathbf{AD}(\mathbf{nD})\right]$$

- Situación posterior

$$\mathbf{F}\left[\mathbf{nF}-1;\ldots\ldots,\mathbf{AF}(\mathbf{jF}-1),\mathbf{KP}\left(\left\lfloor\frac{3*\left\lceil\frac{m}{2}\right\rceil-3}{2}\right\rfloor-\mathbf{nI}\right),\mathbf{AF}(\mathbf{jF}+1),\ldots\ldots\right]$$

$$\mathbf{I}\left[\left\lfloor\frac{3*\left\lceil\frac{m}{2}\right\rceil-3}{2}\right\rfloor;\mathbf{AI}(0),\ldots\ldots,\mathbf{AI}(\mathbf{nI}),\mathbf{KF}(\mathbf{jF}),\mathbf{AP}(0),\ldots\ldots,\mathbf{KP}\left(\left\lfloor\frac{3*\left\lceil\frac{m}{2}\right\rceil-3}{2}\right\rfloor-\mathbf{nI}-1\right),\mathbf{AP}\left(\left\lfloor\frac{3*\left\lceil\frac{m}{2}\right\rceil-3}{2}\right\rfloor-\mathbf{nI}-1\right)\right]$$

$$\mathbf{D}\left[\left\lfloor\frac{3*\left\lceil\frac{m}{2}\right\rceil-3}{2}\right\rfloor;\mathbf{AP}\left(\left\lfloor\frac{3*\left\lceil\frac{m}{2}\right\rceil-3}{2}\right\rfloor-\mathbf{nI}\right),\mathbf{KP}\left(\left\lfloor\frac{3*\left\lceil\frac{m}{2}\right\rceil-3}{2}\right\rfloor-\mathbf{nI}+1\right),\ldots\ldots,\mathbf{AP}(\mathbf{nP}),\mathbf{KF}(\mathbf{jF}+1),\mathbf{AD}(0),\ldots\ldots,\mathbf{AD}(\mathbf{nD})\right]$$

- Procedimiento Recombinacion-3/2

```
Procedimiento Recombinacion-3/2(NodoI, NodoP, NodoD, NodoF, jF)
NodoI: hermano inmediato izquierdo de NodoP (E).
NodoP: nodo bajo mínimos (E).
NodoD: hermano inmediato derecho de NodoP (E).
NodoF: padre de NodoP (E/S).
jF: posición dentro de NodoF del puntero a NodoP (E).
{
  nKI = ⌈(NodoI.nK + NodoP.nK + NodoD.nK + 1)/2.0⌉;
  nKD = ⌊(NodoI.nK + NodoP.nK + NodoD.nK + 1)/2.0⌋;
  nKIDif = nKI – NodoI.nK;
  nKDDif = nKD – NodoD.nK;

  NodoI.nK++;
  NodoI.K[NodoI.nK] = NodoF.K[jF];
  NodoI.A[NodoI.nK] = NodoP.A[0];
  MoverClaves(NodoP, NodoI, 1, nKIDif - 1, IZQUIERDA);
  NodoI.nK = nKI;
```

```
  DesplazarClaves(NodoD, 1, nKDDif, DERECHA);
  NodoD.A[nKDDif] = NodoD.A[0];
  NodoD.K[nKDDif] = NodoF.K[jF + 1];
  NodoD.A[0] = NodoP.A[nKIDif];
  MoverClaves(NodoP, NodoD, nKIDif + 1, nKDDif - 1, DERECHA);
  NodoD.nK = nKD;

  ExtraerClave(NodoF, jF);

  NodoF.K[jF] = NodoP.K[nKIDif];

  ExtraerNodo(NodoP);
  EscribirNodo(NodoI)
  EscribirNodo(NodoD);
  return;
}
```

9. ALGORITMO DE EXTRACCIÓN EN ÁRBOL-B (ROTACIÓN-E/N, RECOMBINACIÓN-3/2)

```
Funcion ExtraerAB2(R, X)
R: raíz del árbol (E/S).
X: clave a extraer (E).
{
  if (!BuscarAMC(R, X, Pila, Top))
     return FALSE;
  NodoP = Pila[Top].Nodo;
  Pos = Pila[Top];
  if (NodoP.A[Pos] != Nulo)
     { T = Top;
       LeerNodo(NodoP.A[Pos], NodoQ);
       Top++;
       Pila[Top].Nodo = NodoQ;
       Pila[Top].Posicion = 0;
       while (NodoQ.A[0] != Nulo)
          { LeerNodo(NodoQ.A[0], NodoQ);
            Top++;
            Pila[Top].Nodo = NodoQ;
            Pila[Top].Posicion = 0;
          }
       Pila[T].Nodo.K[Pos] = NodoP.K[Pos] = NodoQ.K[1];
       EscribirNodo(NodoP);
       Pos = 1;
       NodoP = NodoQ;
     }
  ExtraerClave(NodoP, Pos);
  while ((NodoP.nK < ⌈m/2.0⌉ - 1) && (NodoP.Direccion != R))
     { Top--;
       NodoF = Pila[Top].Nodo;
       jF = Pila[Top].Posicion;
       if (jF < NodoF.nK)
          { LeerNodo(NodoF.A[jF + 1], NodoD);
            if (NodoD.nK > ⌈m/2.0⌉ - 1)
               { DRotacion-E/N(NodoP, NodoF, NodoD, jF);
                 return TRUE;
               }
          }
       if (jF > 0)
          { LeerNodo(NodoF.A[jF - 1], NodoI);
            if (NodoI.nK > ⌈m/2.0⌉ - 1)
               { IRotacion-E/N(NodoI, NodoF, NodoP, jF);
                 return TRUE;
               }
          }
       if ((jF > 0) && (jF < NodoF.nK))
          Recombinacion-3/2(NodoI, NodoP, NodoD, NodoF, jF);
       else if (jF == 0) DRecombinacion-2/1(NodoP, NodoF, NodoD, jF);
       else              IRecombinacion-2/1(NodoI, NodoF, NodoP, jF);
       NodoP = NodoF;
     }
  if (NodoP.nK != 0)
     EscribirNodo(NodoP);
  else
      { R = NodoP.A[0];
        ExtraerNodo(NodoP);
      }
  return TRUE;
}
```