L'organisation en B-tree (Suite)

Opérations sur les B-trees : la recherche

Recherche l'enregistrement de clé k :

1. Commencer à la racine ;

2. Si le noeud courant est une feuille (un bac), trouver l'enregistrement dans ce bac ;

3. Si le noeud courant est un noeud d'index, trouver dans cet index la clé k' qui couvre k. Appliquer l'algorithme de recherche au noeud désigné par le pointeur associé à k'.

```
const
    e = ··· ;  emax = ··· ;  {emax = 2e − 1 ; e > 0}
    d = ··· ;  dmax = ··· ;  {dmax = 2d − 1 ; d > 1}
type
    Btree = ↑Noeud;
    Btree_plus = ↑Noeud_plus;
    Etiquette = (index, feuille);
    Noeud =
       record
       case etiquette : Etiquette of
              index: (
                 nombre: 0..dmax;
                 cle: array[2..dmax] of Cle;
                 fils: array[1..dmax] of btree;
              );
              feuille: (
                 nombre: 0..emax;
                 enreg: array[1..emax] of enregistrement;
              );
       end;
    Noeud_plus = record { utilisé dans l'insertion }
                    plus_petite_cle  : Cle;
                    bt               : btree;
                    end;
```

```
function recherche(c: Cle; bt: Btree; var enreg: Enregistrement): boolean;
var position: integer;
begin

if bt↑.etiquette = feuille

   then if dans_feuille(c, bt, position)
           then begin
                enreg := bt↑.enreg[position];
                recherche := true;
                end
           else recherche := false
   else  recherche :=
         recherche(c, bt↑.fils[pos_dans_index(c, bt)], enreg) ;


end;
```

Opérations sur les B-trees : l'insertion

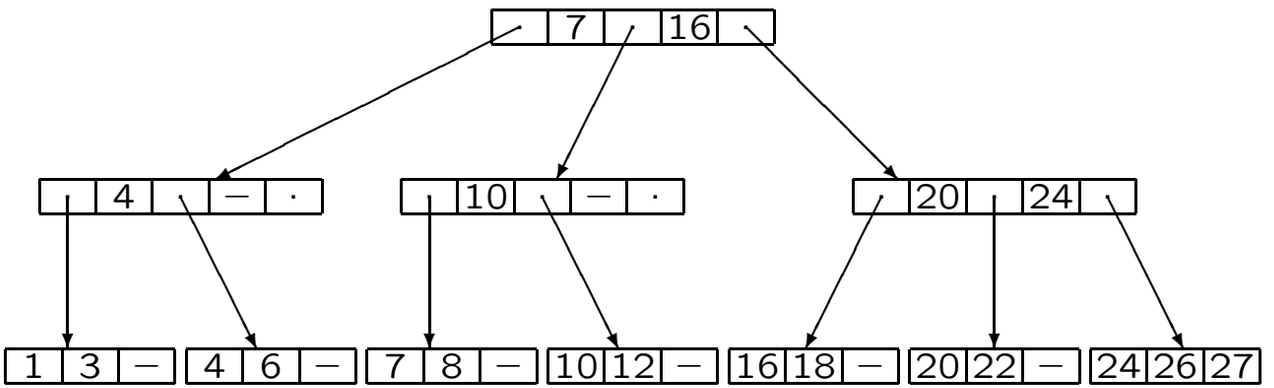
Insertion d'un enregistrement dans un B-tree :

- Si le B-tree est vide :
  - Créer une racine de type feuille,
  - y inserer l'enregistrement.
- Sinon, rechercher la feuille ℓ où insérer l'enregistrement.
  - Si ℓ n'est pas pleine
    - insèrer l'enregistrement en préservant l'ordre.

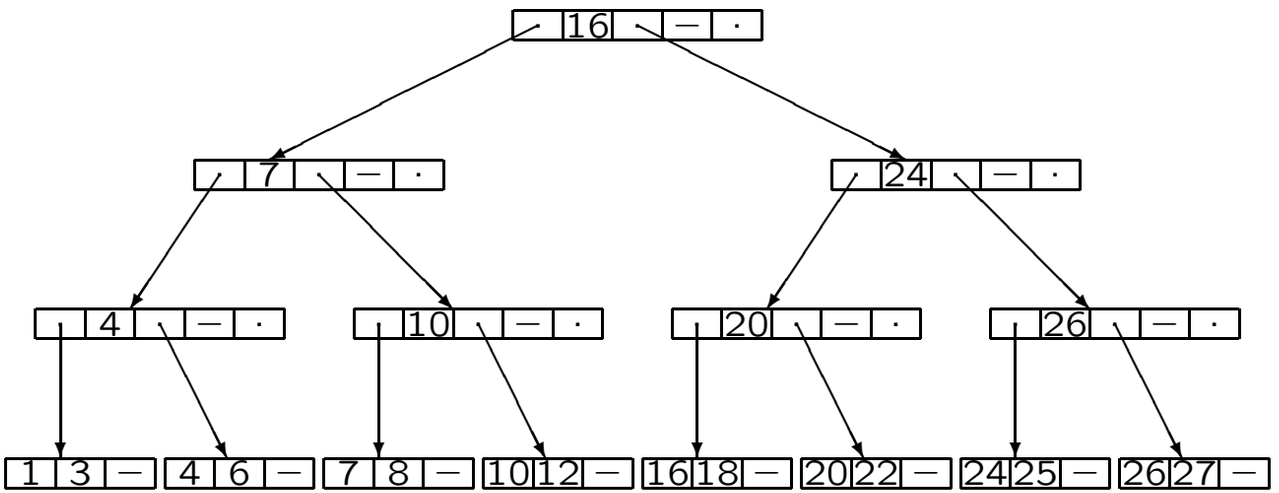
- Si ℓ est pleine
  - créer une nouvelle feuille ℓ',
  - répartir les $2e$ enregistrements entre ℓ et ℓ',
  - mettre à jour le B-tree en remontant dans celui-ci.

  La division peut se propager tant que les ancêtres de ℓ sont pleins. Si on remonte jusqu'à la racine, on la divise en deux et on crée une nouvelle racine avec deux successeurs.

Exemple



Insertion d'un enregistrement de clé 25 . . .


16
7
24
4
10
20
26
1 3
4 6
7 8
10 12
16 18
20 22
24 25
26 27

Algorithme d'insertion

```
procedure inserer(enreg_a_inserer: Enregistrement; var bt: Btree);
var
  bt_plus : Btree_plus;
  racine  : Btree;
begin
if bt = nil
   then begin
        new(bt, feuille);
        with bt↑ do
             begin
             etiquette := feuille; nombre := 1; enreg[1] := enreg_a_inserer;
             end;
        end
   else begin
        bt_plus := inser_noeud(enreg_a_inserer, bt);
        if  bt_plus ≠ nil
           then begin
                new(racine, index);
                with racine↑ do
                     begin
                     etiquette := index;
                     fils[1] := bt; cle[2] := bt_plus↑.plus_petite_cle;
                     fils[2] := bt_plus↑.bt; nombre := 2;
                     end;
                bt := racine;
                end;
        end;

end;
```

```
function  inser_noeud(erg: Enregistrement; bt: Btree): Btree_plus;
var
  position : integer;
  bt_plus  : Btree_plus;

begin

with bt↑ do
      begin
      if etiquette = feuille
         then if dans_feuille(erg.cle,bt,position)
                  then begin
                       erreur(deja_present,erg.cle,bt); inser_noeud := nil;
                       end
                  else inser_noeud := inser_feuille(erg, bt, position)
         else  begin
               position := pos_dans_index(erg.cle,bt);
               bt_plus := inser_noeud(erg,fils[position]);
               if  (bt_plus = nil)
                  then inser_noeud := nil
                  else inser_noeud :=
                       inser_index(bt_plus, bt, position+1) ;
               end;
      end;


end;
```

Opération sur les B-trees : la suppression

Effacement d'un enregistrement de clé k d'un B-tree.

Si le B-tree n'est pas vide.

Soit ℓ la feuille contenant l'enregistrement à supprimer (trouvé par une recherche). Après avoir supprimé l'enregistrement de cette feuille, trois cas peuvent se présenter.

1. La feuille ℓ est la racine du B-tree et elle est devenue vide. On supprime cette feuille et le B-tree devient vide.

2. La feuille ℓ compte au moins e enregistrements. Si la clé k apparaît dans un noeud d'index, mettre l'index à jour.

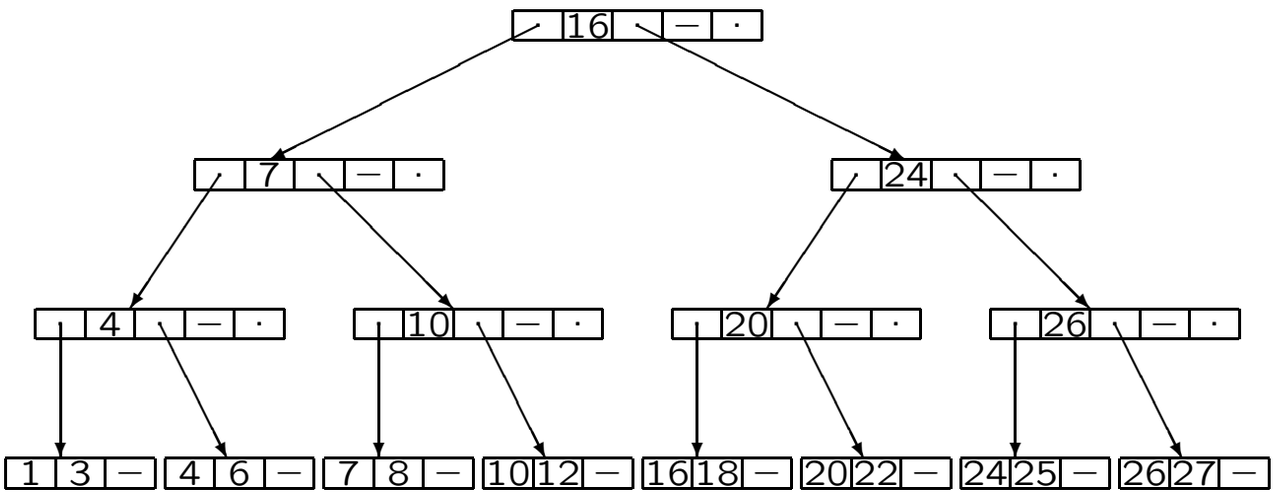
3. La feuille ℓ ne contient que $e-1$ enregistrements. Réorganiser les enregistrements de ℓ et ceux d'une feuille voisine ℓ' (possédant au moins e éléments) d'une des deux manières suivantes.

   - Si ℓ' a plus de e enregistrements, distribuer équitablement les enregistrements entre ℓ et ℓ'.

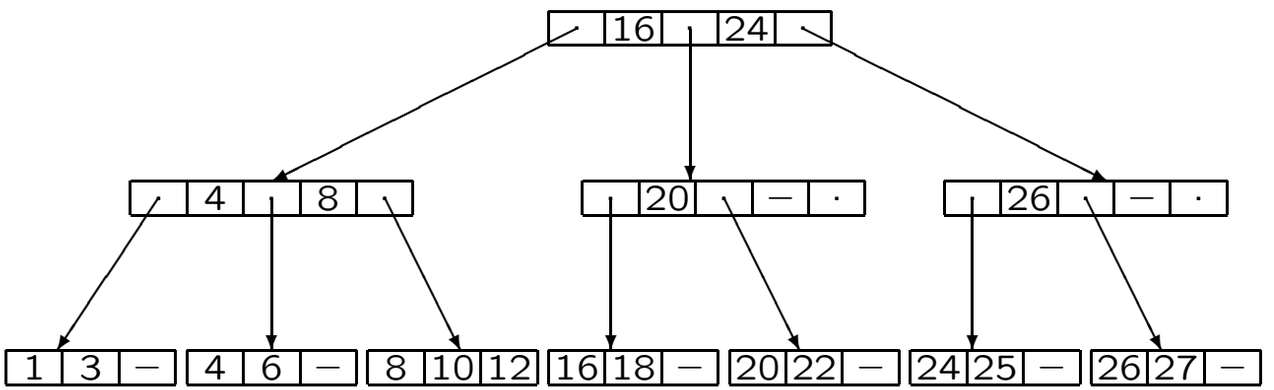
   - Si ℓ' a exactement e éléments, fusionner les enregistrements de ℓ et de ℓ' et supprimer la feuille devenue vide.

   Propager cette mise à jours en remontant vers la racine. Si l'on arrive à la racine du B-tree et que celle-ci ne comporte plus qu'un sous-arbre supprimer la racine, le Btree se réduisant à ce successeur unique.

exemple



Suppression de l'enregistrement de clé 7 . . .


16
24
4
8
20
26
1
3
4
6
8
10
12
16
18
20
22
24
25
26
27

Algorithme de suppression

```
procedure  supprimer(k: Cle; var bt: Btree);
var
  dummy : integer;
  racine   : Btree;

begin
if  bt=nil

   then erreur(arbre_vide,k,bt)
   else  begin
         dummy := supprimer_1(k,bt);
         if  (bt↑.nombre = 0)
            then begin
                 dispose(bt);
                 bt := nil;
                 end
            else  if   (bt↑.etiquette = index) and
                       (bt↑.nombre = 1)
                    then begin
                         racine := bt;
                         bt := bt↑.fils[1];
                         dispose(racine);
                         end;
         end;


end;
```

```
function  supprimer_1 (k: Cle; bt: Btree): Cle;
var
  i : integer;
begin

with bt↑ do
      begin
      if etiquette = feuille
         then begin
                if dans_feuille(k, bt, pos)
                   then begin
                          nombre := nombre - 1;
                          for i := pos to nombre do
                              enreg[i] := enreg[i+1];
                          end
                   else  erreur(non_trouve,k,bt); ;
                supprimer_1 := enreg[1].cle;
                end
         else
                { voir transparent suivant                                    }

                supprimer_1 := supprimer_2(k,bt) ;
      end;


end;
```

```
function  supprimer_2 (k: Cle; bt: Btree): Cle;
var
  i,pos,pos2 : integer;
  cle_min    : Cle;
begin

with bt↑ do
      begin
      pos := pos_dans_index(k, bt);
      cle_min := supprimer_1(k,fils[pos]);
      if pos > 1
         then if (k = cle[pos])
                   then cle[pos] := cle_min ;
      supprimer_2 := cle_min;
      if   ((fils[pos]↑.etiquette = feuille) and (fils[pos]↑.nombre < e))
           or
           ((fils[pos]↑.etiquette = index) and (fils[pos]↑.nombre < d))
         then begin
                 if pos = nombre
                    then pos2 := pos - 1
                    else  pos2 := pos + 1 ;
                 if   ( (fils[pos2]↑.etiquette = feuille) and
                          (fils[pos2]↑.nombre = e))
                        or
                        ( (fils[pos2]↑.etiquette = index) and
                          (fils[pos2]↑.nombre = d))
                    then fusion(bt, min(pos, pos2))
                    else  arrange(bt, min(pos,pos2)) ;
                 end;
      end;


end;
```

Opérations sur les B-trees : la modification

- On ne modifie pas la clé :

  1. rechercher l'enregistrement,
  2. le réécrire.

- On modifie la clé :

  1. suppression
  2. insertion.

Les B-trees :Performances

- choix de la taille des noeuds : 1 bloc.
- Le nombre de lectures de blocs = le nombre de noeuds d'une branche du B-tree
- *longueur d'une branche* = le nombre d'arcs de cette branche
- *longueur d'une branche* +1 = le nombre nombre de noeuds

Calcul de la longueur maximale des branches

- Longueur maximale des branches lorsque les noeuds sont remplis au minimum.

- n enregistrements

- e enregistrements par feuille

$\Rightarrow$ nombre de feuilles :

$$\frac{n}{e}.$$

- nombre de successeurs de chaque noeud intérieur : d

$\Rightarrow$ nombre de noeuds juste au dessus des feuilles (au sommet de chemins de longueur 1 vers les feuilles) :

$$\frac{n}{ed}$$

$\Rightarrow$ nombre de noeuds se trouvant au sommet de chemins de longueur i :

$$\frac{n}{ed^i}$$

- Profondeur du B-tree = le plus petit entier i tel que

$$\frac{n}{ed^{i-1}} < 2d$$

- ou encore tel que

$$d^i > \frac{n}{2e}$$

- donc,

$$i > \log_d \left(\frac{n}{2e} \right)$$

- et,

$$i = \left\lfloor \log_d \left(\frac{n}{2e} \right) \right\rfloor + 1.$$

Nombre de noeuds (blocs) à lire

- pour une recherche :

$$\left\lfloor \log_d \left(\frac{n}{2e} \right) \right\rfloor + 2.$$

- pour une modification :

$$\left\lfloor \log_d \left(\frac{n}{2e} \right) \right\rfloor + 3$$

Exemple

- $e = 3$
- $d = 43$
- 10^7 enregistrements

$\Rightarrow$ nombre maximum d'accès lors d'une recherche :

$$\left\lfloor \log_{43} \left(\frac{10^7}{6} \right) \right\rfloor + 2 = \lfloor 3.81 \rfloor + 2 = 5.$$

(pour un index clairsemé : 23)