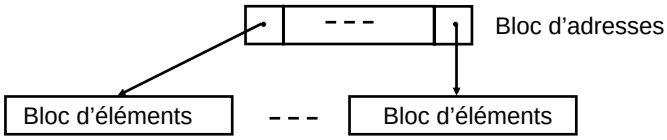
FICHIERS

Mémoire de masse découpée en blocs
Fichier : liste chaînée de blocs, ou
arbre de blocs (répertoires - fichiers)



Blocs de 4096 octets - adresses de 4 octets

⟶ capacité maximale d'un fichier : 4.10^6 octets

OPÉRATIONS DE BASE

Lecture / écriture de blocs - transferts « tampon-bloc »

OPÉRATIONS

AJOUTER un article à un fichier

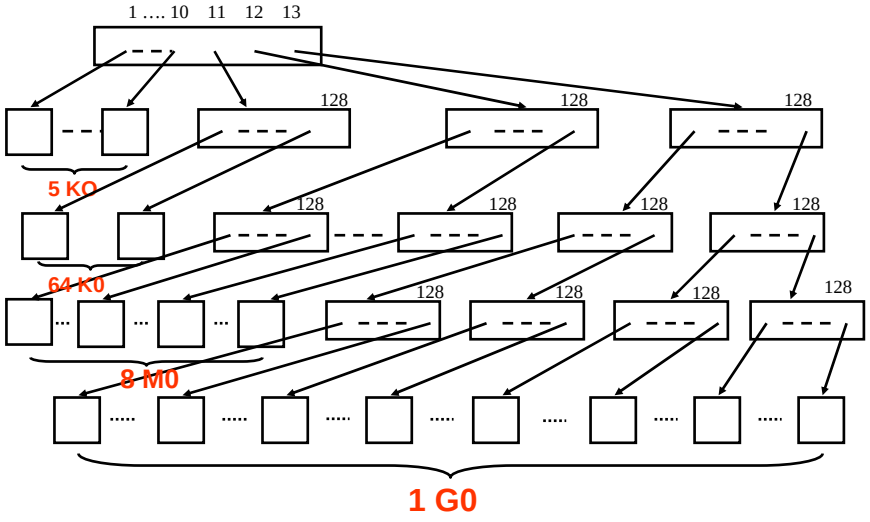
ENLEVER
MODIFIER
RECHERCHER
} Les articles contenant une information donnée dans un fichier

COÛT des opérations en nombre d'accès aux blocs

REPRÉSENTATION des pointeurs :
adresse de bloc + adresse relative dans le bloc

UMLV ©

EXEMPLE D'ORGANISATION SÉQUENTIELLE



et blocs de 512 octets

UMLV ©

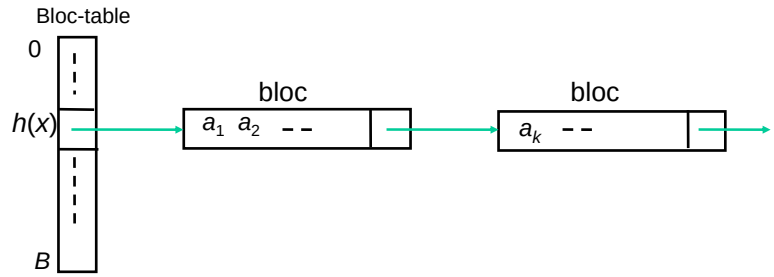
ORGANISATION DES FICHIERS

FICHIERS SÉQUENTIELS

- Liste chaînée de blocs
- Pour suppression d'article :
  article « disponible » ou bit de présence.
- Suite de transformations :
  fichier de mise à jour et algorithme de fusion.

UMLV ©

FICHIERS HACHÉS



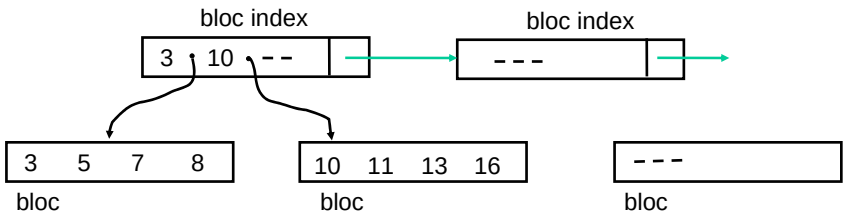
Coût moyen d'une opération : $1 + {}^{n}/_{bB}$ accès
b : nombre d 'articles par bloc

Si *B* bien choisi

RECHERCHE : 2 lectures de bloc
MODIFICATION : 2 lectures de bloc + 1 écriture de bloc

UMLV ©

FICHIERS INDEXÉS



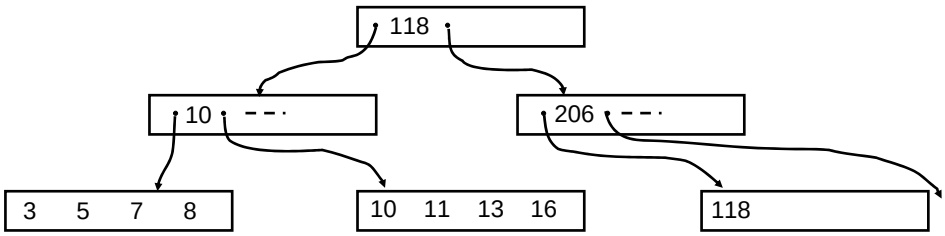
- Articles ordonnés / leur clé
  → recherche dichotomique dans les blocs-index
  → recherche séquentielle dans les blocs d 'articles

COÛT MOYEN d'une opération : $\frac{n}{2b}+1$

- Possibilité d'avoir deux index

UMLV ©

B-ARBRES



- nombre de fils par nœud (≠ racine) dans ($\lceil m/2 \rceil, \ldots, m$)

COÛT MOYEN d'une opération : $1 + \log_{m/2} \lceil n/b \rceil$

UMLV ©

B-arbres

Manipulation d'ensembles avec les opérations

ÉLEMENT (*x*, *A*)
AJOUTER (*x*, *A*)
ENLEVER (*x*, *A*)
} $O(\log_a n)$

Permet la recherche d'intervalles, leur partition, et leur union

Souvent utilisés dans les bases de données
(stockage en mémoire secondaire)

Un **(*a*,*b*) - arbre** est un arbre planaire qui vérifie :

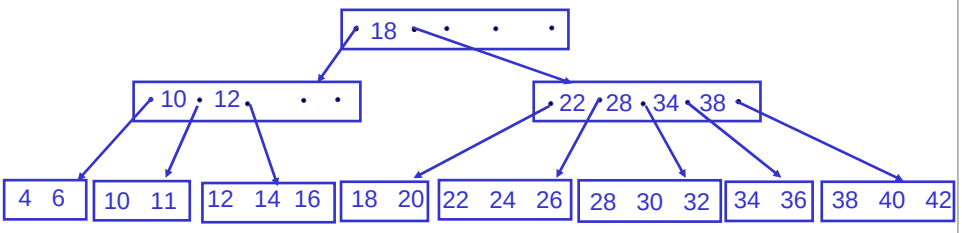
1. La racine est une feuille ou possède au moins 2 fils.
2. Tout nœud interne possède entre ***a*** et ***b*** fils.
3. Toutes les feuilles sont au même niveau

$a \geq 2 \qquad b \geq 2a - 1$

B - arbres $\rightarrow b = 2a - 1$ ($m = b$)
2 - 3 arbres
2 - 3 - 4 arbres $\leftrightarrow$ arbres bicolores

Représentation des ensembles

(3, 5) - arbre pour {4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 23, 30, 32, 34, 36, 38, 40, 42}



Nœuds internes de la forme (a_0, k_1, a_1, k_2, …, k_{b-1}, a_{b-1}) avec

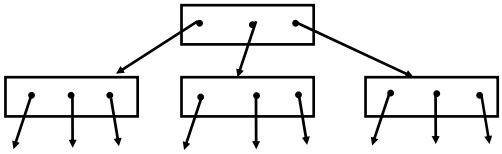
a_i sous-arbre (pointeur sur un bloc)
k_i clé

Propriétés :

$k_1 < k_2 < \ldots < k_b$,
feuilles de $a_{i-1} < k_i \leq$ feuilles de a_i.

TAILLES

UMLV ©



$1 = b^0$

$3 = b^1$

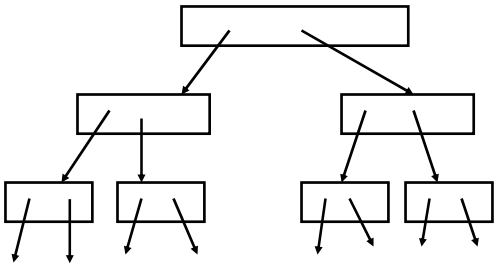
$9 = b^2$

⋮ ⋮

nombre de feuilles $\leq b^{\text{hauteur}}$

hauteur $\geq \log_b$ nombre d'éléments

UMLV ©



1

2 = 2

4 = 2.a

⋮ ⋮

8 = 2.a^2

$a = \lceil m/2 \rceil$

nombre de feuilles $\geq 2.a^{h-1}$ (si $h \geq 1$)

hauteur $\leq 1 + \log_a \frac{1}{2}$ nombre de feuilles

Taux de remplissage des blocs $\geq \frac{2a-1}{2b-1}$

Taux minimum ~ 50 % si $b = 2a\text{-}1$

IMPLANTATION

```
arbre  = ↑ nœud ;
nœud = record
            n : integer ;    { nombre de sous-arbres }
            k  : array [ 1 .. b-1]  of  clé ;
            a :  array [ 0 .. b-1]  of  arbre ;
       end  ;

function  position (x : clé ; A : arbre) : integer ;
{ plus grand i tel que A ↑. k [ i ] ≤ x, }
{ 0 si i n 'existe pas : x < A ↑. k [1] }
```

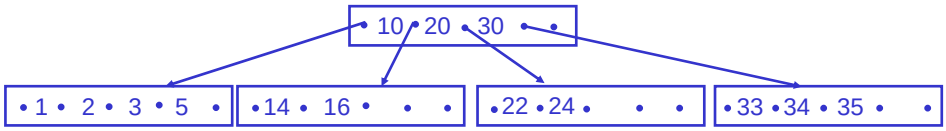


```
function  ELEMENT (x : clé ; A : Arbre) : boolean ;
        var  p : integer ;
begin
        if  (A =  nil )  then
                return  (false)
        else  begin
                p : = position (x, A) ;
                if  (p > 0 )  and  (x = A ↑. k [ p ] ) then
                        return (true)
                else
                        return (ELEMENT (x, A ↑. a [ p ] )) ;
        end  ;
end ;
```

UMLV ©

B-ARBRES de recherche (variante)

Les nœuds internes contiennent des éléments



Nœud interne : (a_0, e_1, a_1, e_2, …e_{m-1}, a_{m-1})

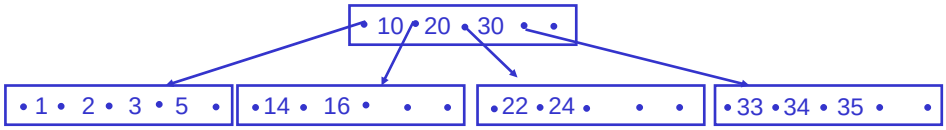
- $e_1 < e_2 < \ldots < e_m$
- éléments de $a_{i-1} < e_i <$ éléments de a_i

- tout nœud (sauf racine) contient entre $\lceil m/2 \rceil$ et m éléments

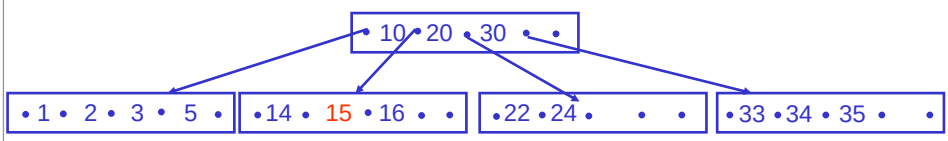
[$a = \lceil m/2 \rceil$, $b = m$]

UMLV ©

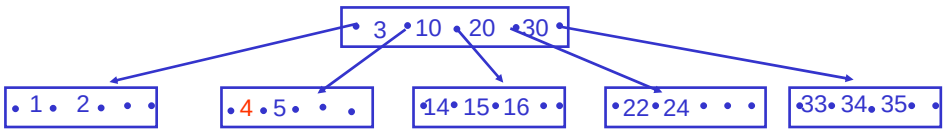
AJOUT D'UN ELEMENT



AJOUT DE 15



AJOUT DE 4



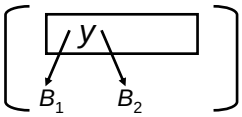
A B-arbre
x élément à ajouter à A

AJOUTER (x, A) = (B_1, y, B_2)

y élément qui a été « chassé » par x ;
B_1, B_2 obtenus par partage éventuel de A.

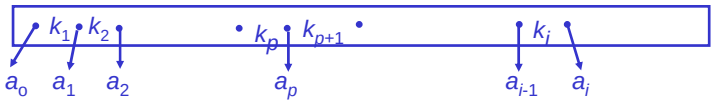
⟶ B_1 et B_2 sont des B-arbres

fonction AJOUT dans ARBRE (x, A) ;
début
(B_1, y, B_2) ← AJOUTER (x, A) ;
si (y indéfini) **alors**
retour (B_1)
sinon
retour
fin ;
fin ;

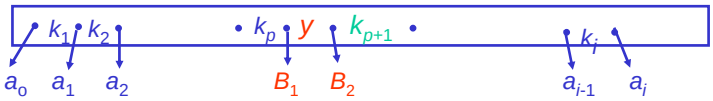


fonction AJOUTER (x, A) ;
début
si (A vide) **alors**
retour (feuille contenant x)
sinon début
p ← position de x dans le bloc-racine de A ;
si (p > 0 et $x = k_p$) **alors** {x déjà dans A}
retour (A, -, ϕ)
sinon début { continuer dans a_p }
(B_1, y, B_2) ← AJOUTER (x, a_p) ;
si (y indéfini) **alors début**
a_p ← B_1 ; retour (A, -, ϕ) ;
fin sinon { y chassé par x dans a_p}
retour (INSERER (B_1, y, B_2, A))
fin
fin
fin ;

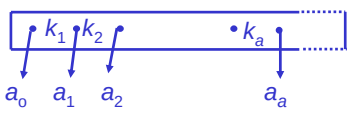
INSERER (B_1, y, B_2, A) {insertion dans un bloc}

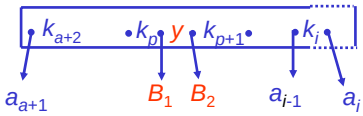


Si ***i < b***



Si ***i = b*** partage du bloc en deux





3 cas, suivant que $p < a$, $p = a$, $p > a$.

A B-arbre
x élément à supprimer

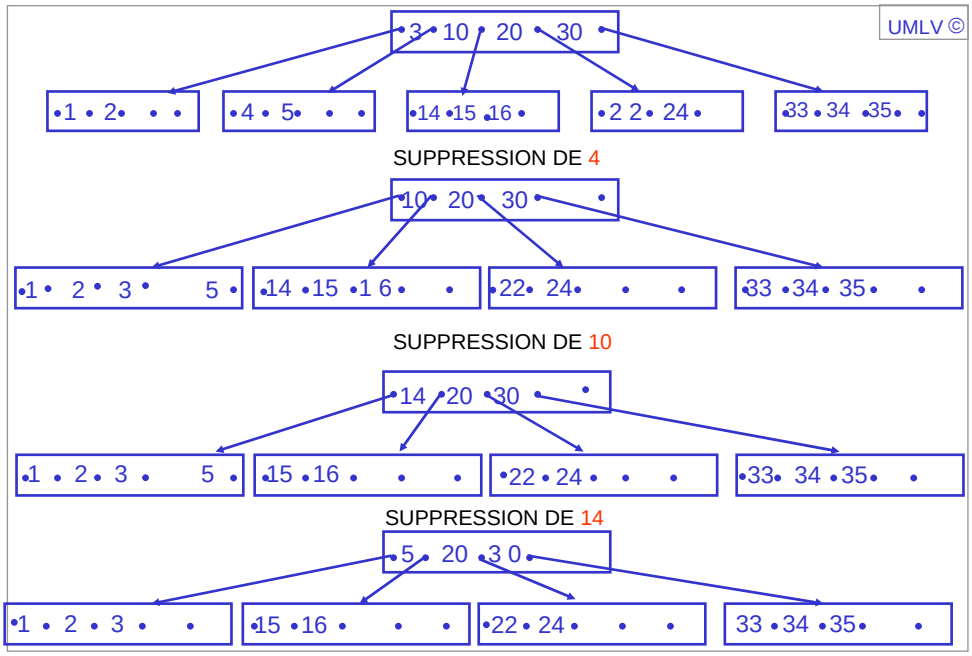
ENLEVER (x, A) = B

B **:** arbre obtenu après suppression de x ;
est un B-arbre si sa racine contient entre
a -1 et b -1 éléments.

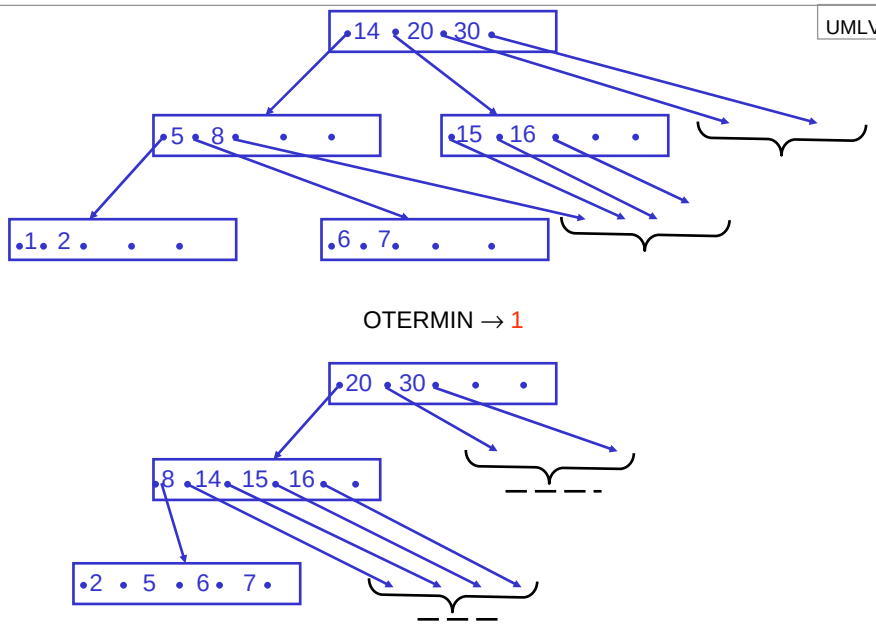
```
fonction  ENLEVER (x, A) ;
début
        A ← ENLEVER (x, A) ;
        si  (A n 'a qu'un fils)  alors
            retour  (fils de A)
        sinon
            retour  (A)
fin  ;
```

UMLV ©

```
fonction  ENLEVER (x, A) ;
début
    si  (A vide)  alors
          retour (A)
    sinon  début
     p ← position de x dans le bloc-racine de A ;
     si  (p = 0 ou k_p ≠ x)  alors début   {on continue dans a_p}
          B ← ENLEVER (x, a_p) ; a_p ← B ;
     fin  sinon  {x = k_p}
     si  (a_p non vide)  alors début
          (y, B) ← OTERMIN (a_p) ;
          k_p ←  y ; a_p ← B ;
     fin  sinon  début       {x = k_p et A est une feuille}
          oter  x du bloc-racine de A ;
          retour  (A) ;
     fin  ;
     si  (le bloc-racine de B contient < a-1 éléments )
     alors   retour  (EQUILIBRER (A, p))
fin  ;
```

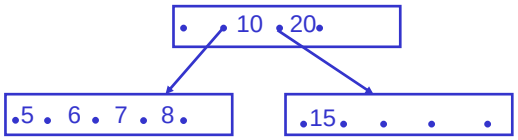
UMLV ©



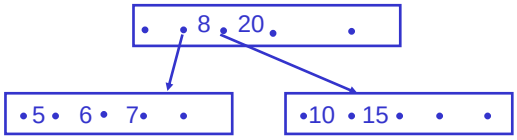
UMLV ©

fonction OTERMIN (*A*) ;
début
  si (la racine est une feuille) **alors début**
    ôter k_1 du bloc-racine de *A* ;
    retour (k_1, *A*) ;
  fin sinon début
    (*y*, *B*) ← OTERMIN (a_o) ; a_o ← *B* ;
    si (le bloc-racine de *B* contient < *a*-1 éléments)
    alors retour (*y*, EQUILIBRER (*A*, 1))
    sinon retour (*y*, *A*) ;
  fin
fin ;

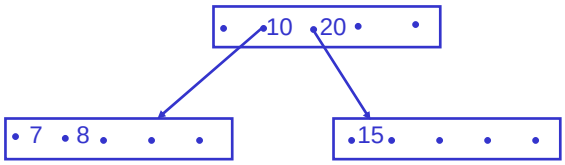
UMLV ©



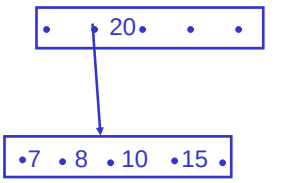
EQUILIBRER



UMLV ©



EQUILIBRER



VARIATIONS

Destinées à éviter ou retarder le partage des blocs.
Assurent un meilleur remplissage des blocs en moyenne.

Pratiquer un ÉQUILIBRAGE, quand un bloc est surchargé,
en examinant :

- le nœud de gauche ou le nœud de droite

ou

- tous les frères du bloc

ou

- tous les descendants du parent du bloc.

On peut ainsi répartir la charge sur tous les nœuds examinés.