

Les B-arbres

Algorithmes et structures de données, 2025-2026

P. Albuquerque (B410) et O. Malaspinas (A401), ISC, HEPIA

2026-05-13

En partie inspiré des supports de cours de P. Albuquerque

Les B-arbres (rappel)

Définition : B-arbre d'ordre n

Les B-arbres (rappel)

Définition : B-arbre d'ordre n

- Chaque page d'un arbre contient au plus $2 \cdot n$ clés;
- Chaque page (excepté la racine) contient au moins n clés;
- Chaque page qui contient m clés contient soit :
 - 0 descendants;
 - $m + 1$ descendants.
- Toutes les pages terminales apparaissent au même niveau.

Les B-arbres (rappel)

Est-ce un B-arbre ?

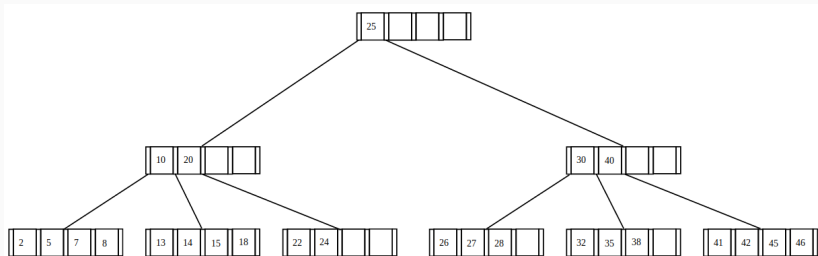


Figure 1 : B-arbre d'ordre 2.

Les B-arbres (rappel)

Est-ce un B-arbre ?

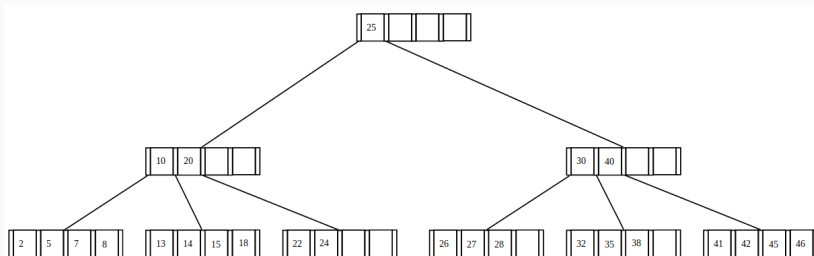


Figure 1 : B-arbre d'ordre 2.

Bien sûr !

Les B-arbres (rappel)

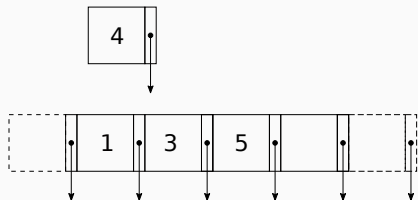
L'algorithme d'insertion

L'algorithme d'insertion

0. Rechercher la feuille (la page a aucun enfant) où insérer ;
1. Si la page n'est pas pleine, insérer dans l'ordre croissant.
2. Si la page est pleine, on sépare la page en son milieu :
 - 2.1 On trouve la médiane, M , de la page ;
 - 2.2 On met les éléments $< M$ dans la page de gauche de M et les $> M$ dans la page de droite de M ;
 - 2.3 M est insérée récursivement dans la page parent.

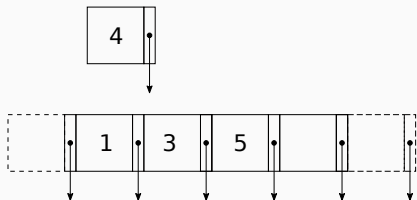
Les B-arbres (rappel)

L'insertion cas nœud pas plein, insertion 4 ?

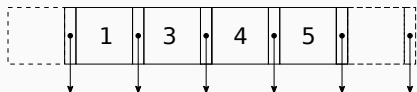


Les B-arbres (rappel)

L'insertion cas nœud pas plein, insertion 4 ?



Solution



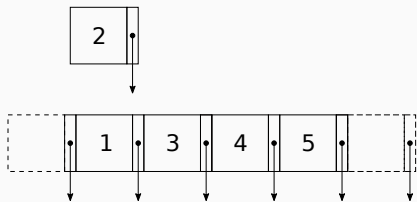
Les B-arbres (rappel)

L'insertion cas nœud pas plein, insertion N

- On décale les éléments plus grand que N ;
- On insère N dans la place “vide” ;
- Comme la page ne déborde pas, on a terminé.

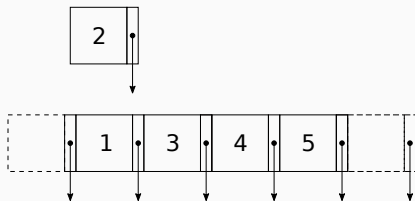
Les B-arbres (rappel)

L'insertion cas nœud plein, insertion 2 ?

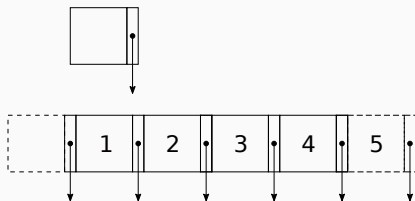


Les B-arbres (rappel)

L'insertion cas nœud plein, insertion 2 ?

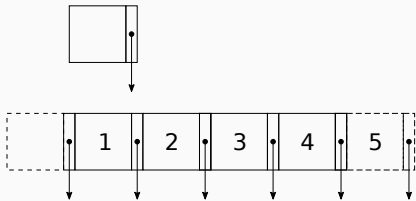


Solution



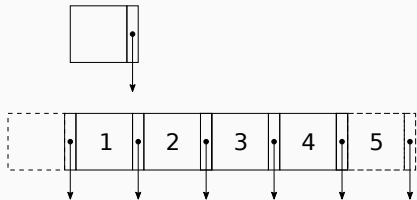
Les B-arbres (rappel)

L'insertion cas nœud plein, promotion 3 ?

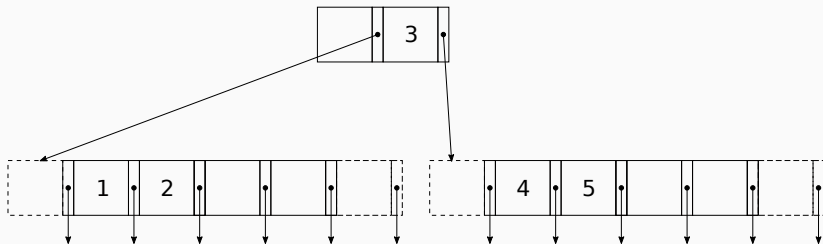


Les B-arbres (rappel)

L'insertion cas nœud plein, promotion 3 ?



Solution



Les B-arbres (rappel)

L'insertion cas nœud plein, insertion N

- On décale les éléments plus grands que N ;
- On insère N dans la place “vide” ;
- Si la page est pleine :
 - On trouve la valeur médiane M de la page (quel indice?) ;
 - On crée une nouvelle page de droite ;
 - On copie les valeurs à droite de M dans la nouvelle page ;
 - On promeut M dans la page du dessus ;
 - On connecte le pointeur de gauche de M et de droite de M avec l'ancienne et la nouvelle page respectivement.

Les B-arbres (rappel)

Pseudo-code structure de données

Les B-arbres (rappel)

Pseudo-code structure de données

```
struct page
    entier ordre, nb
    element tab[2*ordre + 2]
```

```
struct element
    entier clé
    page pg
```

Les B-arbres (rappel)

Les fonctions utilitaires (5min matrix)

```
booléen est_feuille(page)      // la page est-elle une feuille?  
entier position(page, valeur) // à quelle indice insère-t-on?  
booléen est_dans_page(page, valeur) // la valeur est-elle dans la page?
```

Les B-arbres (rappel)

Les fonctions utilitaires (5min matrix)

```
booléen est_feuille(page)      // la page est-elle une feuille?  
entier position(page, valeur) // à quelle indice insère-t-on?  
booléen est_dans_page(page, valeur) // la valeur est-elle dans la page?
```

```
booléen est_feuille(page)  
    retourne (page.tab[0].pg == vide)  
  
entier position(page, valeur)  
    i = 0  
    tant que i < page.nb && valeur >= page.tab[i+1].clef  
        i += 1  
    retourne i
```

```
booléen est_dans_page(page, valeur)  
    i = position(page, valeur)  
    retourne (page.nb > 0 && page.tab[i].val == valeur)
```

Les B-arbres (rappel)

Les fonctions utilitaires

```
page nouvelle_page(ordre) // créer une page
```

Les B-arbres (rappel)

Les fonctions utilitaires

```
page nouvelle_page(ordre)  // créer une page
```

```
page nouvelle_page(ordre)
    page = allouer(page)
    page.ordre = ordre
    page.nb = 0
    page.tab = allouer(2*ordre+2)
    retourner page
```

Les B-arbres (rappel)

Recherche de page

```
page recherche(page, valeur) // retourner la page contenant  
                             // la valeur ou vide
```

Les B-arbres (rappel)

Recherche de page

```
page recherche(page, valeur) // retourner la page contenant  
                               // la valeur ou vide
```

```
page recherche(page, valeur)  
    si est_dans_page(page, valeur)  
        retourne page  
    sinon si est_feuille(page)  
        retourne vide  
    sinon  
        recherche(page.tab[position(page, valeur) - 1],  
                  valeur)
```

Les B-arbres (nouveau)

Les fonctions

```
page inserer_valeur(page, valeur) // insérer une valeur
```


Les B-arbres (nouveau!)

Les fonctions

```
page inserer_valeur(page, valeur) // insérer une valeur
```

```
page inserer_valeur(page, valeur)
    element = nouvel_element(valeur)
    // ici élément est modifié pour savoir
    // s'il faut le remonter
    inserer_element(page, element)
    si element.page != vide && page.nb > 2*page.ordre
        // si on atteint le sommet!
        page = ajouter_niveau(page, element)
    retourne page
```


Les fonctions

```
rien inserer_element(page, element) // insérer un element  
// et voir s'il remonte
```

```
rien inserer_element(page, element)  
  si est_feuille(page)  
    placer(page, element)  
  sinon  
    sous_page =  
      page.tab[position(page, element.clé) - 1].page  
    inserer_element(sous_page, element)  
    // un element a été promu  
    si element.page != vide  
      placer(page, element)
```

Les fonctions (5min matrix)

```
rien placer(page, element) // insérer un element
```

Les fonctions (5min matrix)

```
rien placer(page, element) // insérer un element
```

```
rien placer(page, element)
    pos = position(page, element.clé)
    pour i de 2*page.ordre à pos+1
        page.tab[i+1] = page.tab[i]
    page.tab[pos+1] = element
    page.nb += 1
    si page.nb > 2*page.ordre
        scinder(page, element)
```

Les fonctions (5min matrix)

```
rien scinder(page, element) // casser une page et remonter
```

Les fonctions (5min matrix)

```
rien scinder(page, element) // casser une page et remonter
```

```
rien scinder(page, element)
    nouvelle_page = nouvelle_page(page.ordre)
    nouvelle_page.nb = page.ordre
    pour i de 0 à ordre inclu
        nouvelle_page.tab[i] = page.tab[i+ordre+1]
    element.clé = page.tab[ordre+1].clé
    element.page = nouvelle_page
```


Les fonctions (5min matrix)

```
page ajouter_niveau(page, element) // si on remonte à la  
                                     // racine, on doit créer  
                                     // une nouvelle racine
```

```
page ajouter_niveau(page, element)  
    tmp = nouvelle_page(page.ordre)  
    tmp.tab[0].page = page  
    tmp.tab[1].clé = element.clé  
    tmp.tab[1].page = element.page  
    retourne tmp
```

Les B-arbres : suppression

Cas simplissime

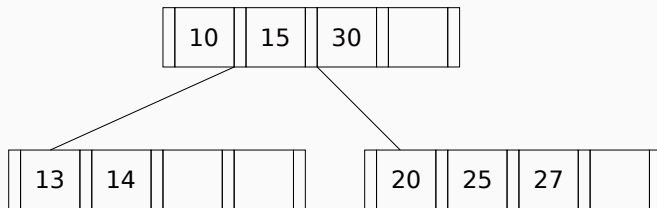


Figure 2 : Suppression de 25.

Les B-arbres : suppression

Cas simplissime

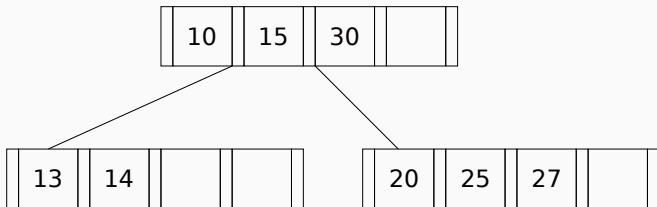


Figure 2 : Suppression de 25.

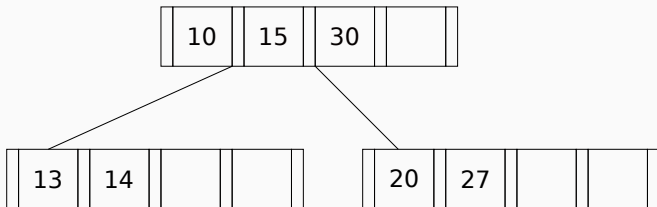


Figure 3 : 25 supprimé, on décale juste 27.

Les B-arbres : suppression

Cas simple

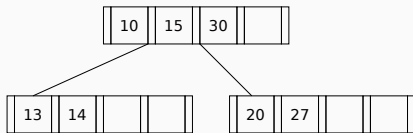


Figure 4 : Suppression de 27.

Les B-arbres : suppression

Cas simple

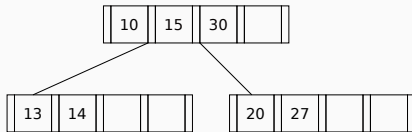


Figure 4 : Suppression de 27.

- On retire 27, mais....
 - Chaque page doit avoir au moins 2 éléments.
 - On doit déplacer des éléments dans une autre feuille ! Mais comment ?

Les B-arbres : suppression

Cas simple

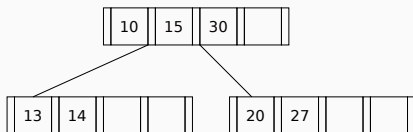


Figure 4 : Suppression de 27.

- On retire 27, mais....
 - Chaque page doit avoir au moins 2 éléments.
 - On doit déplacer des éléments dans une autre feuille ! Mais comment ?

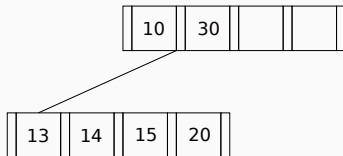


Figure 5 : La médiane de la racine descend, fusion de 20 à gauche, et suppression à droite

Les B-arbres : suppression

Cas moins simple

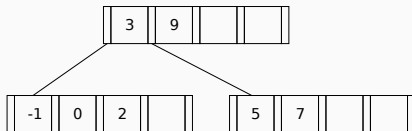


Figure 6 : Suppression de 5.

Les B-arbres : suppression

Cas moins simple

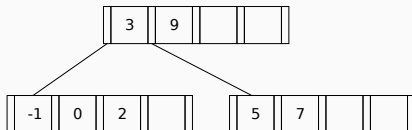


Figure 6 : Suppression de 5.

- Un élément à droite, comment on fait ?
 - Remonter 7, serait ok si racine, mais... ce n'est pas forcément le cas.
 - On redistribue les feuilles.

Les B-arbres : suppression

Cas moins simple

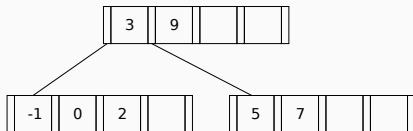


Figure 6 : Suppression de 5.

- Un élément à droite, comment on fait ?
 - Remonter 7, serait ok si racine, mais... ce n'est pas forcément le cas.
 - On redistribue les feuilles.

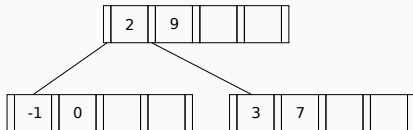


Figure 7 : Descente de 3, remontée médiane des feuilles 2.

Les B-arbres : suppression

Cas ultra moins simple

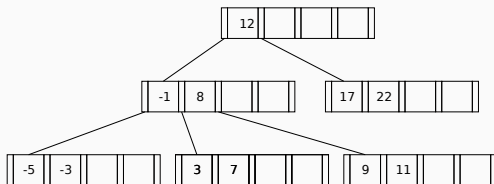


Figure 8 : Suppression de 3.

Les B-arbres : suppression

Cas ultra moins simple

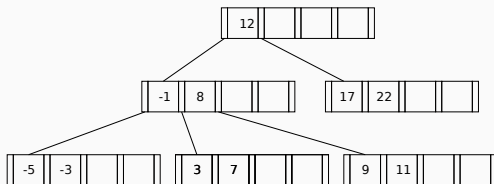


Figure 8 : Suppression de 3.

- 7 seul :
 - Fusionner les feuilles et redistribuer, comment ?

Les B-arbres : suppression

Cas ultra moins simple

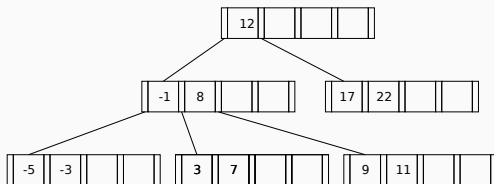


Figure 8 : Suppression de 3.

- 7 seul :
 - Fusionner les feuilles et redistribuer, comment ?

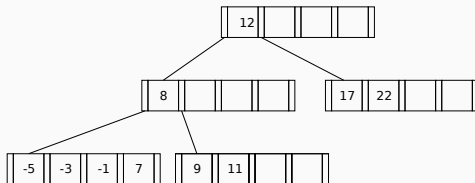


Figure 9 : Descendre -1, déplacer 7 à gauche, et décaler les éléments de droite

Les B-arbres : suppression

Cas ultra moins simple

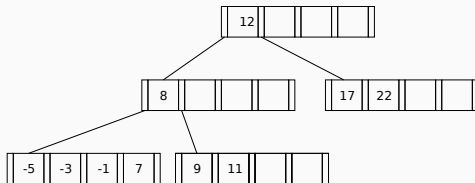


Figure 10 : On a pas fini...

Les B-arbres : suppression

Cas ultra moins simple

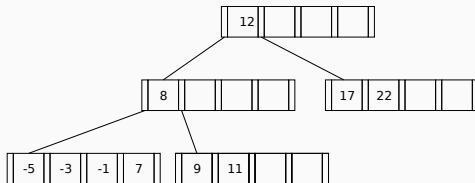


Figure 10 : On a pas fini...

- 8 est seul, ce n'est plus un B-arbre :
 - Fusionner le niveau 2 et redistribuer, comment ?

Les B-arbres : suppression

Cas ultra moins simple

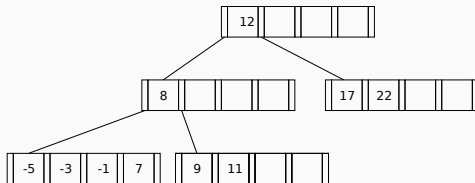


Figure 10 : On a pas fini...

- 8 est seul, ce n'est plus un B-arbre :
 - Fusionner le niveau 2 et redistribuer, comment ?

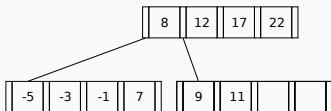


Figure 11 : Fusionner 8, 17, 22 et descendre 12.

Les B-arbres : suppression

Cas ultra moins simple

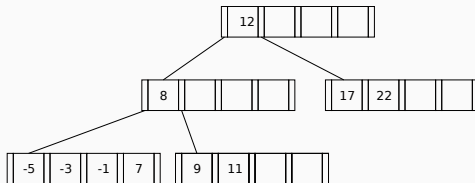


Figure 10 : On a pas fini...

- 8 est seul, ce n'est plus un B-arbre :
 - Fusionner le niveau 2 et redistribuer, comment ?

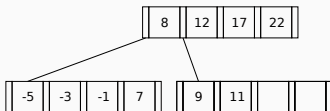


Figure 11 : Fusionner 8, 17, 22 et descendre 12.

- La profondeur a diminué de 1.

Algorithme pour les feuilles !

- Si la clé est supprimée d'une feuille :
 - Si on a toujours n (ordre de l'arbre) clés dans la feuille, on décale simplement les clés.
 - Sinon on combine (récursivement) avec le nœud voisin et on descend la clé médiane.

Les B-arbres : suppression

Cas non-feuille !

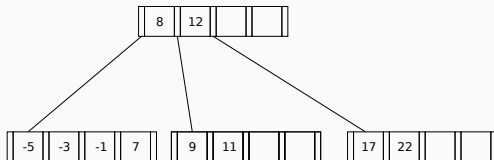


Figure 12 : Suppression de 8.

Les B-arbres : suppression

Cas non-feuille !

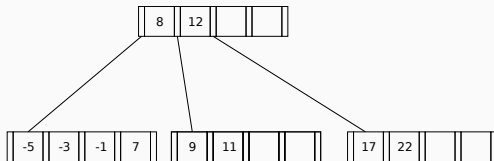


Figure 12 : Suppression de 8.

- On sait comment effacer une valeur d'une feuille, donc ?

Les B-arbres : suppression

Cas non-feuille !

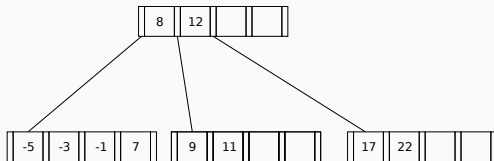


Figure 12 : Suppression de 8.

- On sait comment effacer une valeur d'une feuille, donc ?

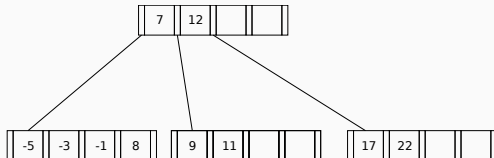


Figure 13 : Échanger le 8 avec le plus grand du sous-arbre de gauche.

- Ensuite ?

Les B-arbres : suppression

Cas non-feuille !

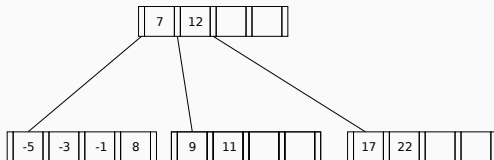


Figure 14 : Suppression de 8.

Les B-arbres : suppression

Cas non-feuille !

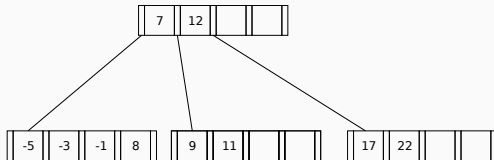


Figure 14 : Suppression de 8.

- On sait comment effacer une valeur d'une feuille !

Les B-arbres : suppression

Cas non-feuille !

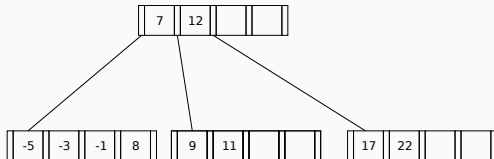


Figure 14 : Suppression de 8.

- On sait comment effacer une valeur d'une feuille !

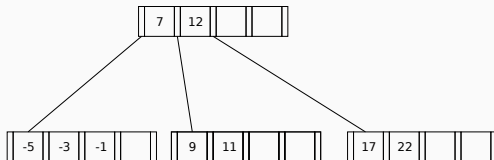


Figure 15 : Yaka enlever le 8 de la feuille comme avant !

Les B-arbres : suppression

Algorithme pour les non-feuilles !

- Si la clé est supprimée d'une page qui n'est pas une feuille :
 - On échange la valeur avec la valeur de droite de la page de gauche.
 - On supprime comme pour une feuille !

Et maintenant des exercices par millions !