

Les B-arbres

Algorithmes et structures de données, 2025-2026

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

2026-05-06

En partie inspiré des supports de cours de P. Albuquerque

Les B-arbres

Les B-arbres

Problématique

- Grands jeux de données (en 1970).
- Stockage dans un arbre, mais l'arbre ne tient pas en mémoire.
- Regrouper les sous-arbres en **pages** qui tiennent en mémoire.

Exemple

- 100 nœuds par page et l'arbre comporte 10^6 nœuds :
 - Recherche B-arbre : $\log_{100}(10^6) = 3$;
 - Recherche ABR : $\log_2(10^6) = 20$.
- Si on doit lire depuis le disque : 10ms par recherche+lecture :
 - 30ms (lecture beaucoup plus rapide que recherche) vs 200ms = 0.2s.

Remarques

- On ne sait pas ce que veut dire B : Bayer, Boeing, Balanced ?
- Variante plus récente B+-arbres.

Les B-arbres

Illustration, arbre divisé en pages de 3 nœuds

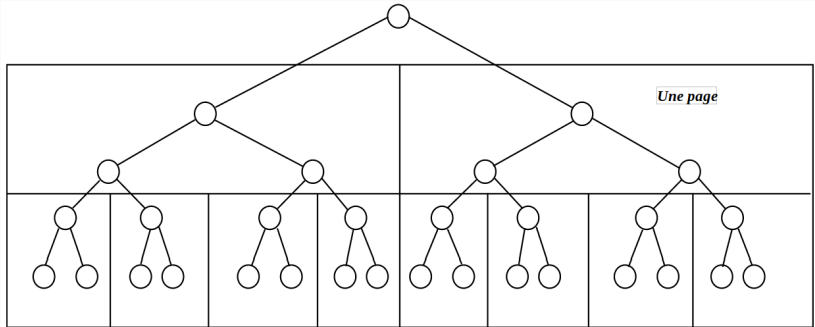


Figure 1 : Arbre divisé en pages de 3 nœuds

Les B-arbres

Illustration, arbre divisé en pages de 3 nœuds

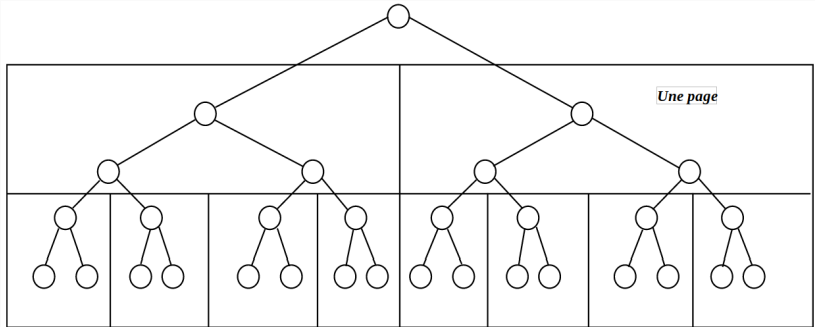


Figure 1 : Arbre divisé en pages de 3 nœuds

Utilisation

- Bases de données (souvent très grandes donc sur le disque) ;
- Systèmes de fichiers.

Avantages

- Arbres moins profonds ;
- Diminution des opérations de rééquilibrage ;
- Complexité toujours en $\log(N)$;

Avantages

- Arbres moins profonds ;
- Diminution des opérations de rééquilibrage ;
- Complexité toujours en $\log(N)$;

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

Est-ce un B-arbre ?

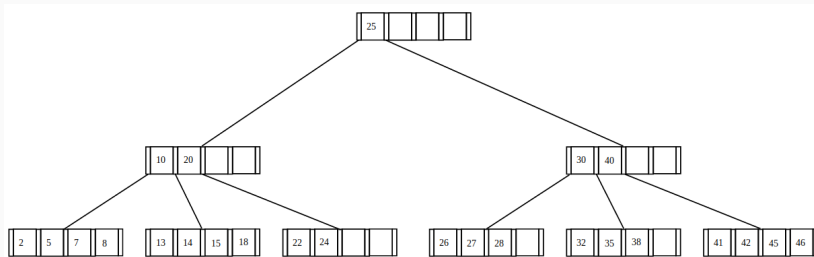


Figure 2 : B-arbre d'ordre 2.

Les B-arbres

Est-ce un B-arbre ?

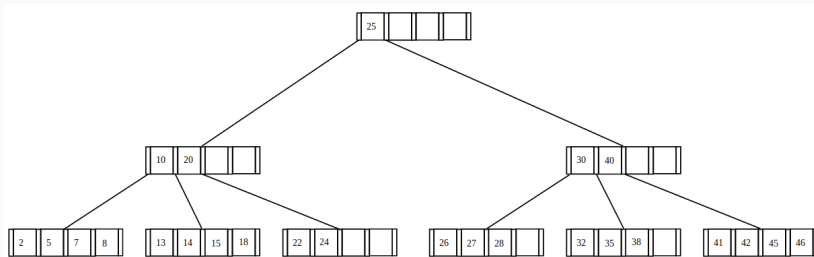


Figure 2 : B-arbre d'ordre 2.

Oui !

- Dans chaque nœud les clés sont **triées**.
- Chaque page contient au moins n clés (sauf à la page racine) et au plus $2n$ clés : check ;
- Chaque nœud avec m clés a $m + 1$ descendants ;
- Toutes les feuilles apparaissent au même niveau.

Les B-arbres

Exemple de recherche : trouver 32

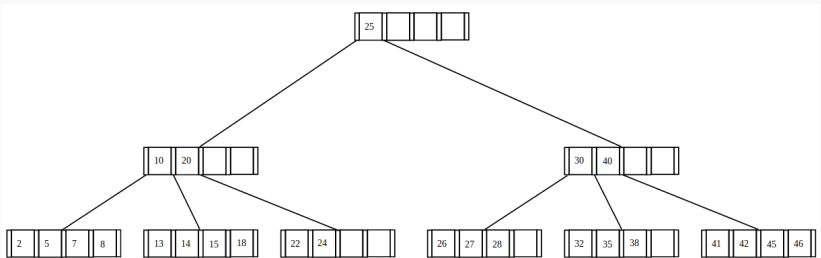


Figure 3 : B-arbre d'ordre 2.

Les B-arbres

Exemple de recherche : trouver 32

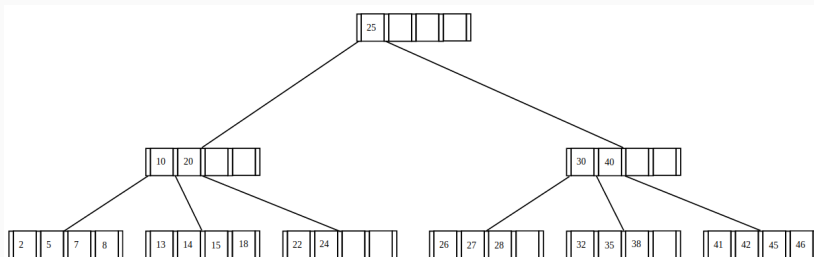


Figure 3 : B-arbre d'ordre 2.

- Si C plus petit que la 1ère clé ou plus grand que la dernière descendre.
- Sinon parcourir (par bisection ou séquentiellement) jusqu'à trouver où descendre entre 2 éléments.

Algorithme de recherche de la clé C

0. En partant de la racine.
1. Si on est dans une feuille :
 - Si C est dans la page, retourner la page ;
 - Sinon c'est perdu.
2. Sinon :
 - Tant que $C < \text{clé}(\text{page})$ passer à la clé suivante
 - Si C est dans la page, retourner la page ;
 - Sinon descendre

Les B-arbres

Disclaimer

- Inspiration de <https://en.wikipedia.org/wiki/B-tree>

Exemples d'insertion : 1

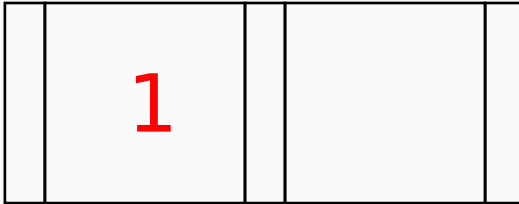


Figure 4 : B-arbre d'ordre 1.

Disclaimer

- Inspiration de <https://en.wikipedia.org/wiki/B-tree>

Exemples d'insertion : 1

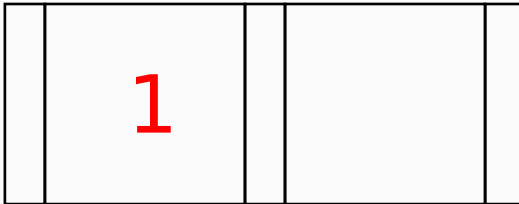


Figure 4 : B-arbre d'ordre 1.

- L'arbre est vide, on insère juste dans la première page.

Exemples d'insertion : 2

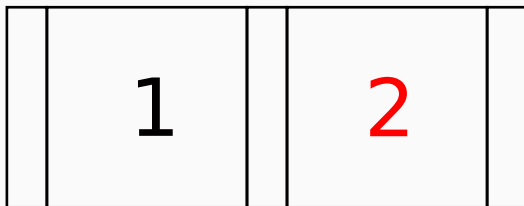


Figure 5 : B-arbre d'ordre 1. Nombre pages max = 2.

Exemples d'insertion : 2

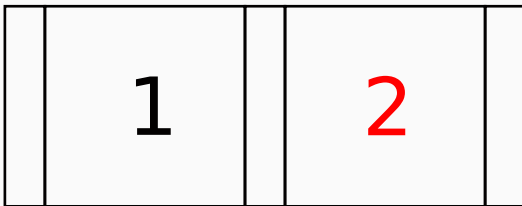


Figure 5 : B-arbre d'ordre 1. Nombre pages max = 2.

- La première page n'est pas pleine, on insère dans l'ordre (après 1).

Exemples d'insertion : 3



Figure 6 : B-arbre d'ordre 1.

- Comment on insère (1min de réflexion avant de donner une réponse!)?

Exemples d'insertion : 3

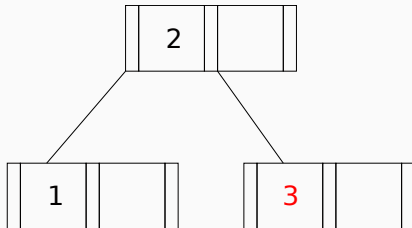


Figure 7 : B-arbre d'ordre 1. Nombre pages max = 2.

Exemples d'insertion : 3

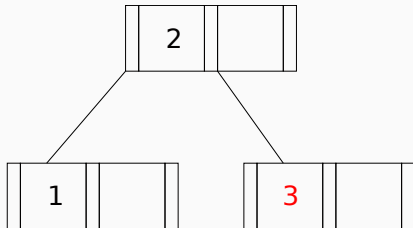


Figure 7 : B-arbre d'ordre 1. Nombre pages max = 2.

- La page est pleine, on crée deux enfants.
- On choisit, 2, la médiane de 1, 2, 3 et il est inséré à la racine.
- 1 descend à gauche, 3 descend à droite.

Exemples d'insertion : 4

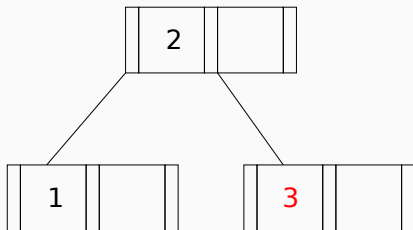


Figure 8 : B-arbre d'ordre 1.

- Comment on insère (1min de réflexion avant de donner une réponse!) ?

Exemples d'insertion : 4

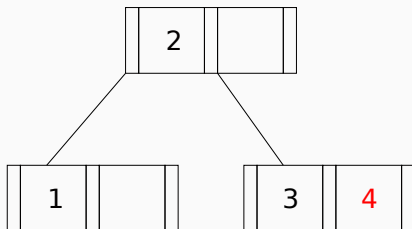


Figure 9 : B-arbre d'ordre 1. Nombre enfants 0 ou 2.

Exemples d'insertion : 4

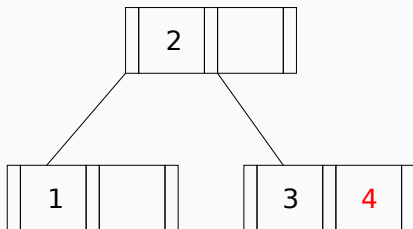


Figure 9 : B-arbre d'ordre 1. Nombre enfants 0 ou 2.

- On pourrait insérer à droite de 2, mais... ça ferait 2 parents pour 2 enfants (mais m parents $\Rightarrow m+1$ enfants ou 0);
- On descend à droite ($4 > 2$);
- On insère à droite de 3.

Exemples d'insertion : 5

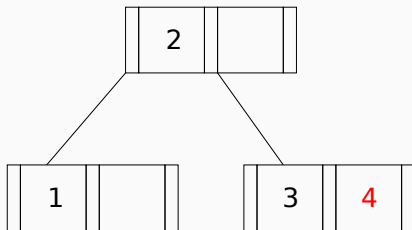


Figure 10 : B-arbre d'ordre 1.

- Comment on insère (1min de réflexion avant de donner une réponse!) ?

Les B-arbres

Exemples d'insertion : 5

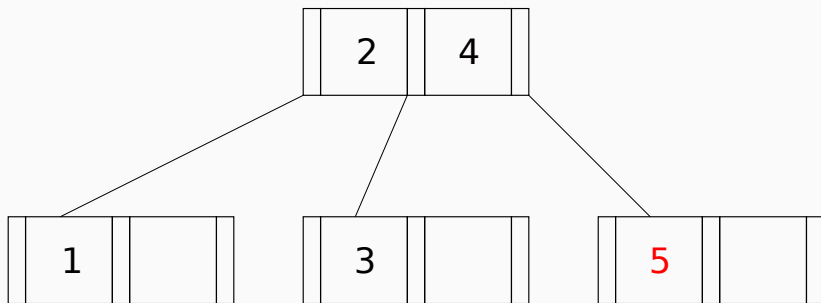


Figure 11 : B-arbre d'ordre 1.

Les B-arbres

Exemples d'insertion : 5

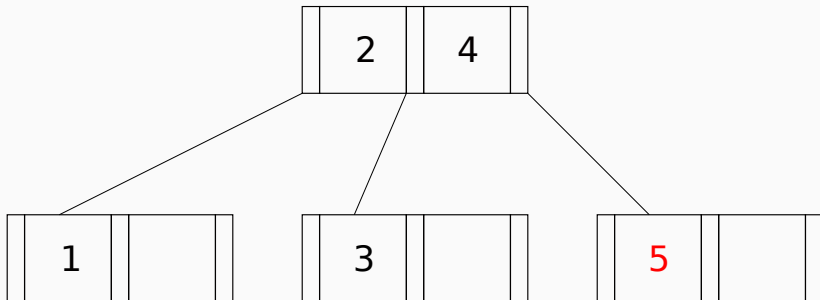


Figure 11 : B-arbre d'ordre 1.

- On descend à droite (on ne peut pas insérer à la racine comme pour 4);
- On dépasse la capacité de l'enfant droite;
- 4, médiane de 3, 4, 5, remonte à la racine;

Exemples d'insertion : 6

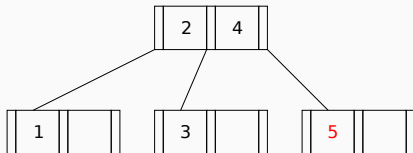


Figure 12 : B-arbre d'ordre 1.

- Comment on insère (1min de réflexion avant de donner une réponse!) ?

Exemples d'insertion : 6

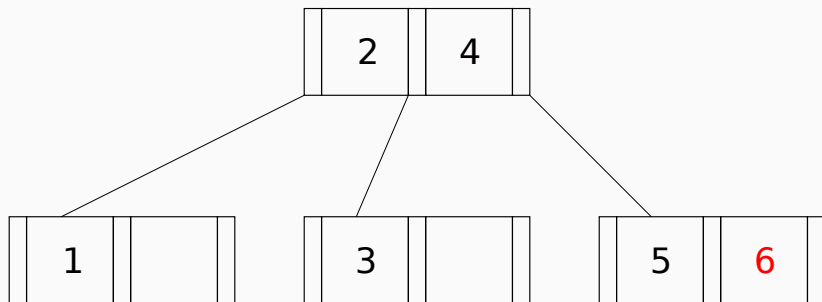


Figure 13 : B-arbre d'ordre 1.

Exemples d'insertion : 6

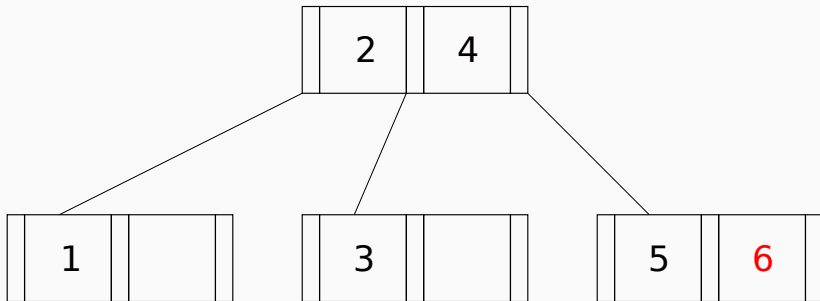


Figure 13 : B-arbre d'ordre 1.

- $6 > 4$ on descend à droite ;
- $6 > 5$ et on a à la place à droite, on insère.

Exemples d'insertion : 7

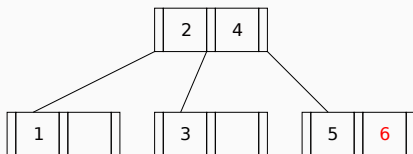


Figure 14 : B-arbre d'ordre 1.

- Comment on insère (1min de réflexion avant de donner une réponse!) ?

Les B-arbres

Exemples d'insertion : 7

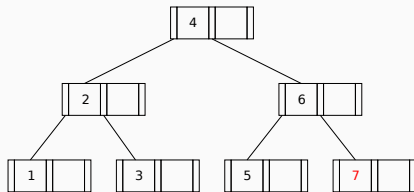


Figure 15 : B-arbre d'ordre 1.

Exemples d'insertion : 7

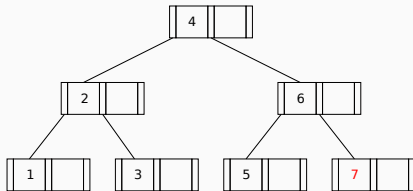


Figure 15 : B-arbre d'ordre 1.

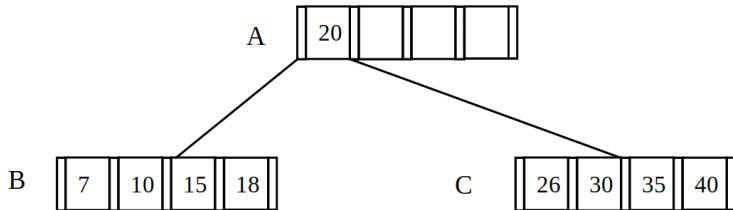
- $7 > 4$ on descend à droite ;
- $7 > 6$ mais on a dépassé la capacité ;
- 6 est la médiane de 5, 6, 7, remonte à la racine ;
- 5 reste à gauche, 7 à droite, mais 6 fait dépasser la capacité de la racine ;
- 4 est la médiane de 2, 4, 6, 4 remonte, 2 reste à gauche, 6 à droite.

L'algorithme d'insertion

0. Rechercher la feuille (la page n'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.

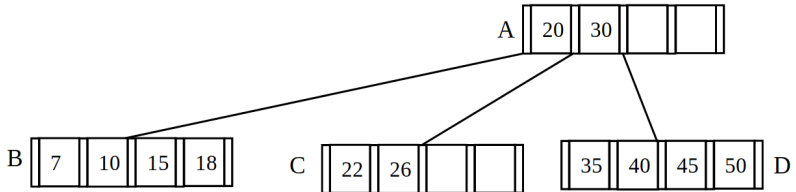
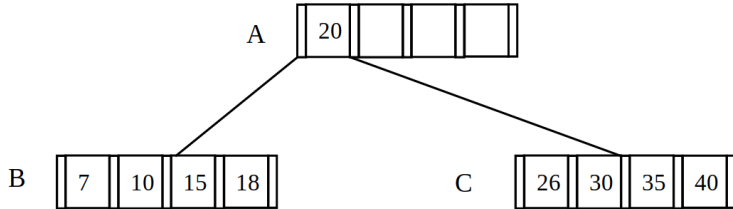
Les B-arbres

Exercice : insérer 22, 45, 50 dans l'arbre d'ordre 2 (3min matrix)



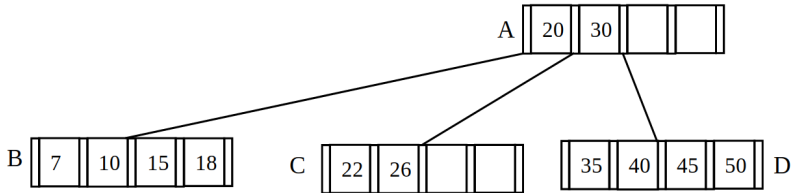
Les B-arbres

Exercice : insérer 22, 45, 50 dans l'arbre d'ordre 2 (3min matrix)



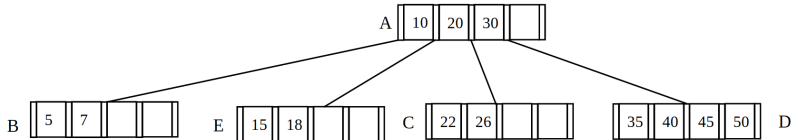
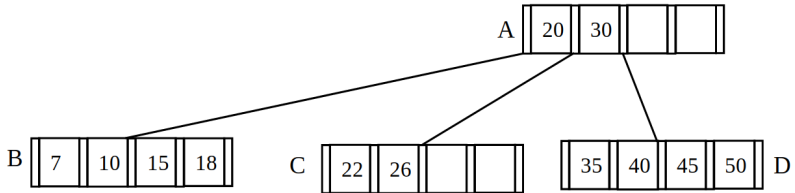
Les B-arbres

Exercice : insérer 5 dans l'arbre d'ordre 2 (3min matrix)



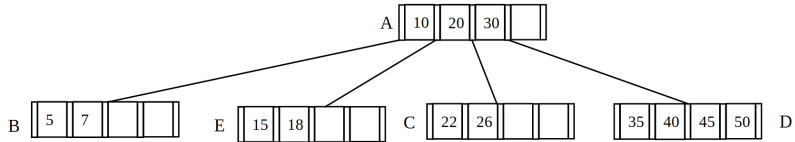
Les B-arbres

Exercice : insérer 5 dans l'arbre d'ordre 2 (3min matrix)



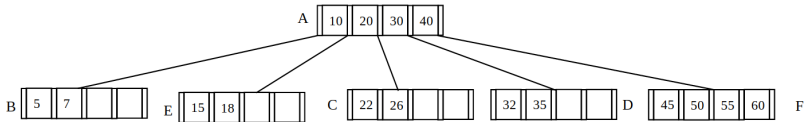
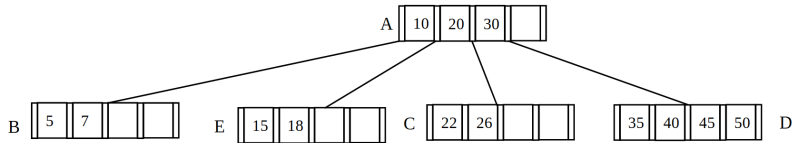
Les B-arbres

Exercice : insérer 32, 55, 60 dans l'arbre d'ordre 2 (3min matrix)



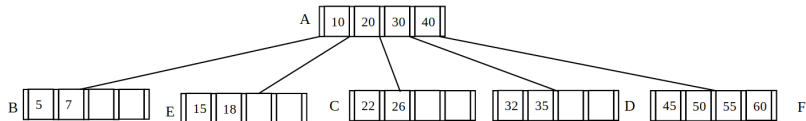
Les B-arbres

Exercice : insérer 32, 55, 60 dans l'arbre d'ordre 2 (3min matrix)



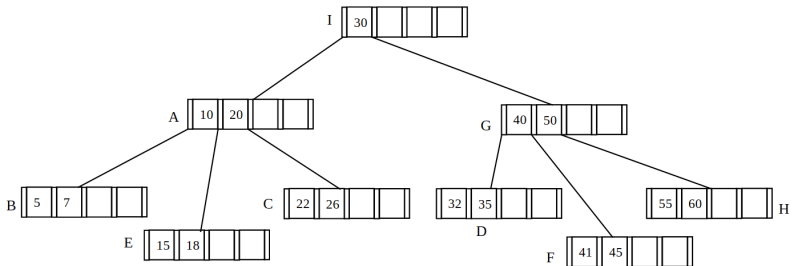
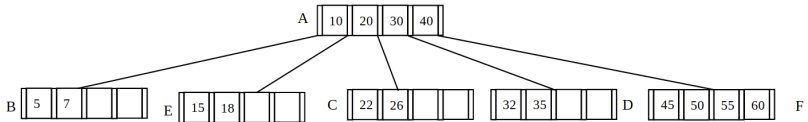
Les B-arbres

Exercice : insérer 41 dans l'arbre d'ordre 2 (3min matrix)



Les B-arbres

Exercice : insérer 41 dans l'arbre d'ordre 2 (3min matrix)



Exercice (matrix, 15min)

- Insérer 20, 40, 10, 30, 15, 35, 7, 26, 18, 22, 5, 42, 13, 46, 27, 8, 32, 38, 24, 45, 25, 2, 14, 28, 32, 41,
- Dans un B-arbre d'ordre 2.

Structure de données

- Chaque page a une contrainte de remplissage, par rapport à l'ordre de l'arbre ;
- Un nœud (page) est composé d'un tableau de clés/pointeurs vers les enfants ;

$P_0 \mid K_1 \mid P_1 \mid K_2 \mid \dots \mid P_i \mid K_{\{i+1\}} \mid \dots \mid P_{\{m-1\}} \mid K_m \mid P_m$

- P_0, \dots, P_m pointeurs vers enfants ;
- K_1, \dots, K_m les clés.
- Il y a $m+1$ pointeurs mais m clés.
- Comment faire pour gérer l'insertion ?

Faire un dessin de la structure de données (3min matrix) ?

Faire un dessin de la structure de données (3min matrix) ?

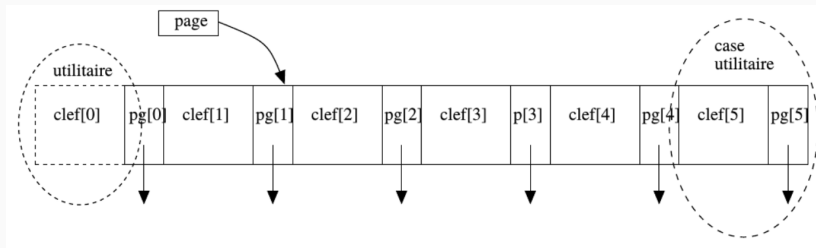
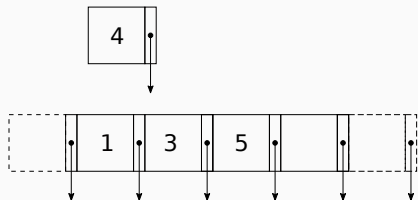


Figure 16 : Structure d'une page de B-arbre d'ordre 2.

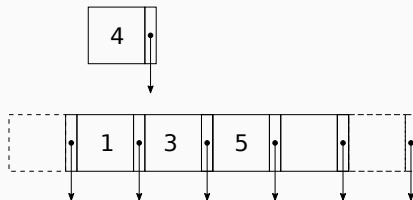
1. On veut un tableau de P_i , $K_i \Rightarrow$ struct ;
2. K_0 va être en "trop" ;
3. Pour simplifier l'insertion dans une page, on ajoute un élément de plus.

L'insertion cas nœud pas plein, insertion 4 ?

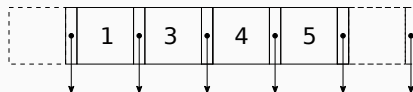


Les B-arbres

L'insertion cas nœud pas plein, insertion 4 ?



Solution

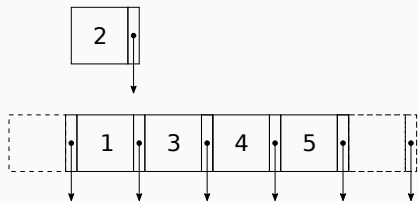


L'insertion cas nœud pas plein, insertion N

- On décale les éléments plus grand que N ;
- On insère N dans la place “vide” ;
- Si la page n'est pas pleine, on a terminé.

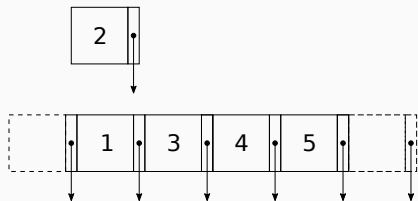
Les B-arbres

L'insertion cas nœud plein, insertion 2 ?

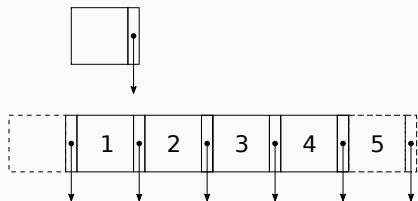


Les B-arbres

L'insertion cas nœud plein, insertion 2 ?

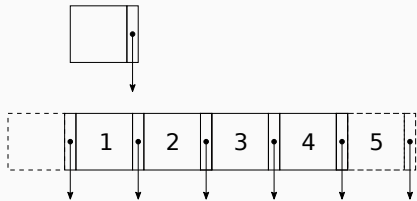


Solution



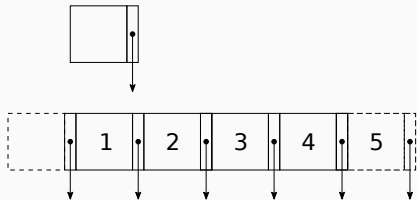
Les B-arbres

L'insertion cas nœud plein, promotion 3 ?

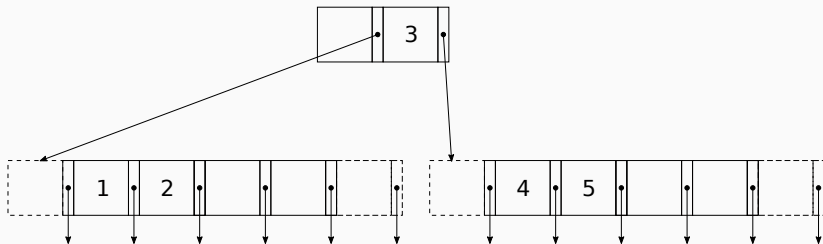


Les B-arbres

L'insertion cas nœud plein, promotion 3 ?



Solution



L'insertion cas nœud plein, insertion N

- On décale les éléments plus grand 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.

Pseudo-code structure de données (3min, matrix) ?

Pseudo-code structure de données (3min, matrix) ?

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

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

Les fonctions utilitaires (5min matrix)

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

Les fonctions utilitaires (5min matrix)

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

Les fonctions utilitaires (5min matrix)

```
page nouvelle_page(ordre) // créer une page  
rien liberer_memoire(page) // libérer tout un arbre!
```

Les B-arbres

Les fonctions utilitaires (5min matrix)

```
page nouvelle_page(ordre)  // créer une page
rien liberer_memoire(page) // libérer tout un arbre!
```

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

rien liberer_memoire(page)
    si est_feuille(page)
        liberer(page.tab)
        liberer(page)
    sinon
        pour fille dans page.tab
            liberer_memoire(fille)
        liberer(page.tab)
        liberer(page)
```

Les fonctions (5min matrix)

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

Les fonctions (5min matrix)

```
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)
```