STD : BTrees

Harmonisation informatique

Bonjour !

Des arbres généraux de recherche équilibrés

ÉCOLE D'INGENIEURS EN INFORMATIQUE

Quelles utilisations pour des BTrees ?

Un BTree est particulièrement approprié pour des systèmes de stockages qui lisent et écrivent des bloc de données assez volumineux, par exemples des bases de données ou des systèmes de fichiers.

Dans les *filesystems* :

- Apple's **HFS+** and **APFS**
- Microsoft's **NTFS**
- some Linux filesystems, such as **Btrfs** and **ext4**

utilisent les Btrees.

Arbres généraux de recherche équilibrés

Définition de la structure

- → Arbres généraux ✅
- → de recherche ❌
- → équilibrés ❌

Arbres généraux de recherche équilibrés

Définition de la structure

- → Arbres généraux ✅
- → de recherche 💡
- → équilibrés ❌

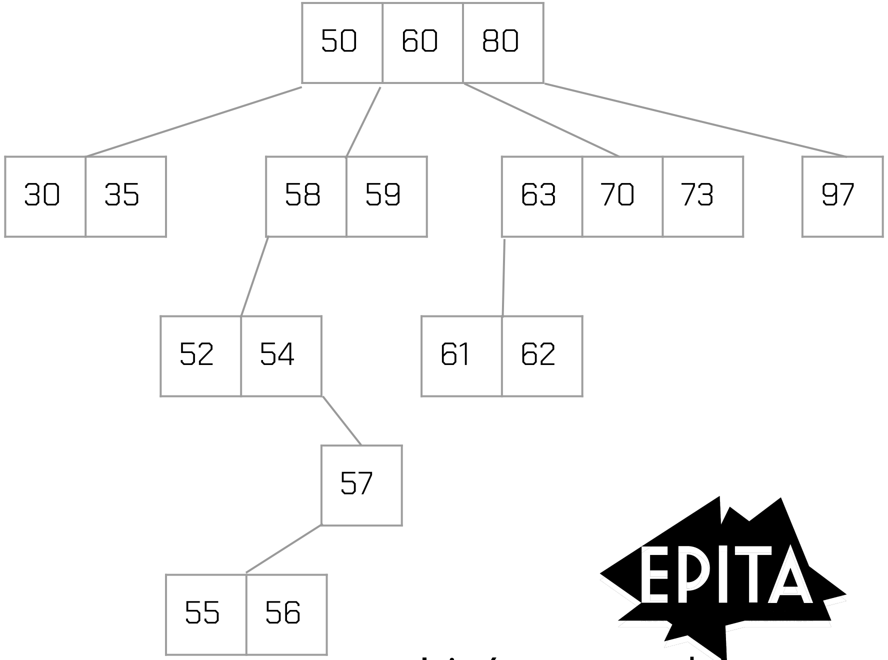
M-way search tree

Arbre (général) de recherche (*m*-way search tree) :

- Chaque nœud (dit *k*-nœud) contient $k - 1$ clés et *k* fils s'il est interne.
- Les clés sont en ordre strictement croissant dans un même nœud.
- Pour chaque clé *x*, son fils gauche contient les clés strictement inférieures à *x*, son fils droit les clés strictement supérieures à *x*.

Contrairement à l'arbre général, un arbre de recherche pourra être vide.

Arbre général de recherche



Ici : 4-way search tree

Arbres généraux de recherche équilibrés

Définition de la structure

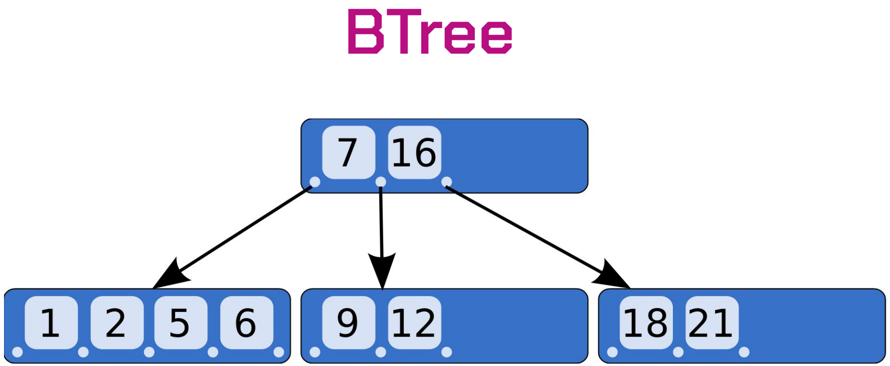
- → Arbres généraux ✅
- → de recherche ✅
- → équilibrés 💡

ÉCOLE D'INGENIEURS EN INFORMATIQUE

Un B-arbre est un arbre général de recherche pour lequel :

- ➔ Toutes les feuilles ont la même profondeur.
- ➔ Il existe t tel que pour chaque k-nœud de l’arbre : $t \leq k \leq 2t$; sauf pour la racine, pour laquelle $2 \leq k \leq 2t$.

  t est appelé le degré minimal (ou ordre) du B-arbre.



Ordre de ce B-arbre : 3

EPITA

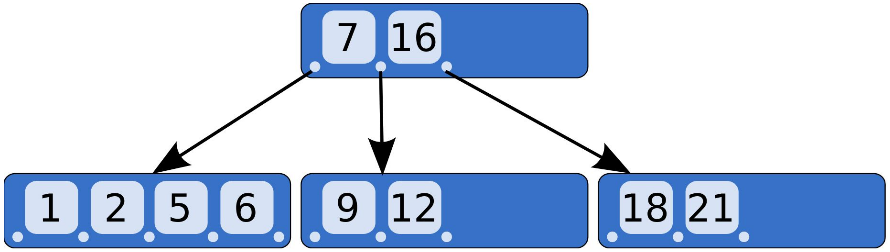
Un B-arbre est un arbre général de recherche pour lequel :

- ➔ Toutes les feuilles ont la même profondeur.
- ➔ Il existe t tel que pour chaque k-nœud de l'arbre : $t \leq k \leq 2t$; sauf pour la racine, pour laquelle $2 \leq k \leq 2t$.

  t est appelé le degré minimal (ou ordre) du B-arbre. Ainsi :

- ➔ Tout nœud autre que la racine doit contenir au moins $t-1$ clés. Tout nœud interne autre que la racine possède au moins t fils. Si l'arbre n'est pas vide, la racine doit posséder au moins une clé.
- ➔ Tout nœud peut contenir au plus $2t - 1$ clés. Un nœud interne peut donc posséder au plus $2t$ fils. Un nœud est complet s'il contient exactement $2t - 1$ clés.

BTree



Ordre de ce B-arbre : 3

EPITA

Arbres généraux de recherche équilibrés

Définition de la structure

- → Arbres généraux ✅
- → de recherche ✅
- → équilibrés ✅

Ajout dans un BTree

Insérer un élément dans un BTree

Où peut-on insérer un élément dans un BTree ?

- → En racine ?
- → N'importe quel nœud où il y a de la place ?
- → En feuille ?

Insérer un élément dans un BTree

Où peut-on insérer un élément dans un BTree ?

- → ~~En racine ?~~
- → ~~N'importe quel nœud où il y a de la place ?~~
- → En feuille ? ✅ (s'il y a de la place)

EPITA
ÉCOLE D'INGENIEURS EN INFORMATIQUE

Une animation pour comprendre

https://www.cs.usfca.edu/~galles/visualization/BTree.html

Credits to David Galles,

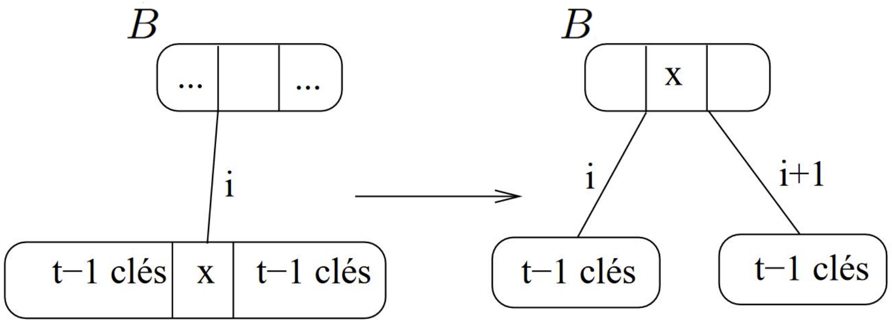
Associate Professor

UNIVERSITY OF SAN FRANCISCO
CHANGE THE WORLD FROM HERE

EPITA

Opération d'éclatement

Quand un nœud risque d'être plein, il faut faire de la place, au cas où on en aurait besoin.



Spécifications :

La procédure `eclate (B, i)` éclate le fils n°i du B-arbre B).

— L'arbre B existe (est non vide) et sa racine n'est pas un $2t$-nœud.

— Le fils i de B existe et sa racine est un $2t$-nœud.

Et la complexité ?

Suppression dans un BTree

Supprimer un élément dans un BTree

Où peut-on *facilement* supprimer un élément dans un BTree ?

- En racine ?
- N'importe quel nœud où il y a suffisamment d'éléments ?
- En feuille ?

Supprimer un élément dans un BTree

Où peut-on *facilement* supprimer un élément dans un BTree ?

- ➔ ~~En racine ?~~
- ➔ ~~N'importe quel nœud où il y a suffisamment d'éléments ?~~
- ➔ En feuille ? ✅ (s'il y a suffisamment d'éléments)

Une animation pour comprendre

https://www.cs.usfca.edu/~galles/visualization/BTree.html

Credits to David Galles,

Associate Professor

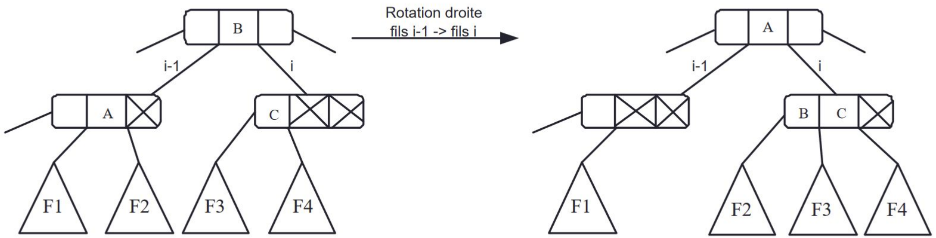
UNIVERSITY OF SAN FRANCISCO

CHANGE THE WORLD FROM HERE

EPITA

Rotation droite

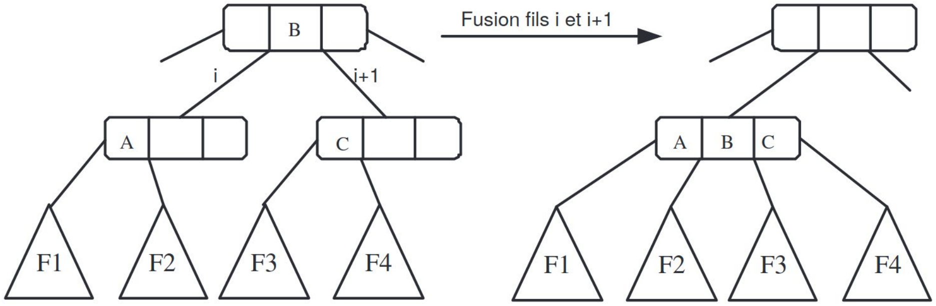
Quand un nœud risque d'être trop peu rempli, il faut ajouter un élément, au cas où on en aurait besoin.



La procédure `rd_gen(B, i)` effectue une rotation du fils $i-1$ vers le fils i (voir figure 6). Conditions : l'arbre B existe, son fils i existe et sa racine n'est pas un $2t$-nœud, le fils $i-1$ existe et sa racine n'est pas un t-nœud.

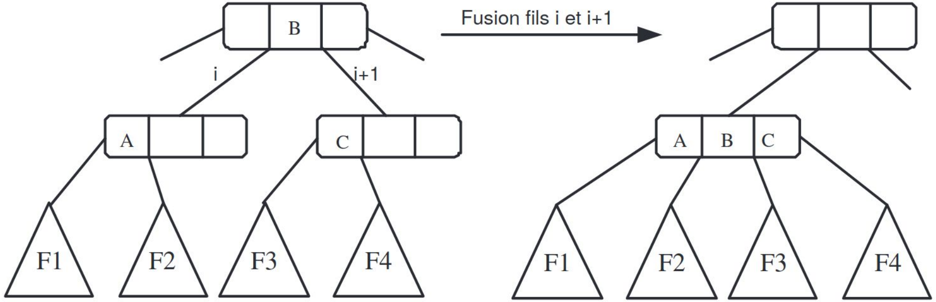
Fusion

Quand on ne peut plus faire de rotation, ni gauche, ni droite, on fusionne.



La procédure `fusion(`B`,` i`)` fusionne les fils i et $i+1$ de l'arbre B (voir figure 7). Conditions : l'arbre B existe et sa racine n'est pas un t-nœud, ses fils i et $i+1$ existent et leurs racines sont des t-nœuds.

Quand on ne peut plus faire de rotation, ni gauche, ni droite, on fusionne.



La procédure `fusion(`B`, `i`)` fusionne les fils i et $i+1$ de l'arbre B (voir figure 7). Conditions : l'arbre B existe et sa racine n'est pas un t-nœud, ses fils i et $i+1$ existent et leurs racines sont des t-nœuds.

Et la complexité ?

On sait gérer les BTrees maintenant.

La suite en TD !