

# Arbres (a,b), B-arbres et recherche externe

# Recherche externe

- On veut implémenter un dictionnaire ordonné dont le nombre d'éléments est trop grand pour la mémoire principale de notre ordinateur
- On va utiliser une mémoire externe pour garder l'information (disque, zip, etc.)
  - ▣ l'information dans la mémoire externe est gardée dans des **blocks B**
  - ▣ On appelle **transfert d'information** le transfert d'un block entre la mémoire externe et la mémoire principale
- Le but de l'implémentation est de réduire le nombre de transferts d'information nécessaire pour chercher, insérer ou supprimer un élément du dictionnaire
- La complexité en temps de l'algorithme de recherche externe sera calculée en fonction du nombre de transferts d'information

# Arbres (a,b)

○ Un arbre  $(a,b)$  est une généralisation d'un arbre (2,4) dans le sens suivant:

- **Nombre d'enfants:** tout noeud interne a au moins  $a$  fils (sauf possiblement la racine) et au plus  $b$  fils.
- **Propriété de profondeur:** tous les noeuds externes ont la même profondeur
- **Spécifications pour a et b:**  $a$  et  $b$  sont des entiers, tels que

$$2 \leq a \leq \frac{(b+1)}{2}$$

○ On va voir qu'en choisissant les entiers  $a$  et  $b$  correctement (en fonction de la taille des blocks mémoires) on va obtenir une structure de donnée qui est efficace en terme de recherche externe

○ **Proposition:** La hauteur d'un arbre  $(a,b)$  est  $\Omega(\log n / \log b)$  et  $O(\log n / \log a)$

# Insertion, débordement et fractionnement

- L'insertion dans un arbre  $(a,b)$  est similaire à l'insertion dans un arbre  $(2,4)$ .
- Insérer un élément dans un  $b$ -noeud  $n$  cause un débordement (on a maintenant un  $b+1$  noeud)
  - ▣ La clé médiane du noeud  $n$  est insérée dans le parent  $u$  du noeud (une nouvelle racine peut être créée)
  - ▣ Le noeud  $n$  est remplacé par un  $\lceil (b+1)/2 \rceil$ - noeud  $n'$  et un  $\lfloor (b+1)/2 \rfloor$ - noeud  $n''$
- Le débordement peut se propager dans le parent  $u$

# Suppression, noeud interne vide, fusion et transfert

- La suppression dans un arbre  $(a,b)$  est similaire à la suppression dans un arbre  $(2,4)$ .
- Supprimer un élément dans un  $a$ -noeud  $n$  cause un sous-chargement du noeud (on a maintenant un  $(a-1)$ -noeud)
  - Si un des frère adjacent n'est pas un  $a$ -noeud, on exécute un transfert
  - Sinon, on exécute une fusion d'un  $a$ -noeud avec le noeud  $n$  et on obtient un nouveau  $(2a-1)$  - noeud
  - C'est la raison pour laquelle  $a \leq (b+1)/2$  dans un arbre  $(a,b)$
- Le problème de sous-chargement peut se propager vers la racine lorsqu'on exécute une fusion

# B-arbres

- Un B-arbre d'ordre  $d$  est un arbre  $(a,b)$  avec

$$a = \lceil \frac{d}{2} \rceil \quad \text{et} \quad b = d$$

- On choisit  $d$  de sorte que la taille des blocks  $B$  de la mémoire externe soit suffisante pour garder en mémoire  $d-1$  entrées et les références aux  $d$  fils d'un noeud, i.e  $d$  est  $\Theta(B)$
- La hauteur de l'arbre est en  $O(\log n / \log a) \stackrel{\text{ici}}{=} O(\log_2 n / \log_2 B) = \log_B n$