TD Algo2 – session 5 – B-Arbres

18 février 2025

Objectifs d'apprentissage :
— manipuler la structure de B-arbre et étudier les implications des propriétés de ce type d'arbres ;
— concevoir des algorithmes relatifs aux B-arbres ;
— mettre en pratique des techniques de preuve pour la validité ou la complexité d'algorithmes.

Les 5 premiers exercices sont essentiels.

Exercice 1 : Énumération

Donner tous les B-arbres valides de degré minimal $t = 2$ permettant de représenter $\{1, 2, 3, 4, 5\}$.

Exercice 2 : Nombre de clés

Pour un degré minimal t, quel est le nombre maximal de clés qui peuvent être stockées dans un B-arbre de hauteur h ?

On a au maximum $2t - 1$ clé par nœud. Au niveau 1, on a 1 nœud. Au niveau 2, on en a au maximum $2t$. Au niveau 3, $4t^2$, etc.
On a donc

$$n \leq (2t-1)\sum_{i=0}^{h}(2t)^i = (2t-1)\frac{(2t)^{h+1}-1}{2t-1} = (2t)^{h+1} - 1$$

Donc, $\log_{2t}(n+1) - 1 \leq h$.

Exercice 3 : Minimum

Écrire l'algorithme qui trouve la clé minimale stockée dans un B-arbre.

On peut soit écrire une version itérative, soit une version récursive. La différence se situera dans l'appel initiale. Pour une versions récursive, on utilisera B-Arbre-Minimum-rec($T.racine$) car la méthode s'applique sur un nœud. Pour la version itérative, on appelle la méthode sur l'arbre directement. Ce principe s'applique pour d'autres méthodes sur les structures d'arbre.

B-Arbre-Minimum(T)

```
x ← T.racine
si x.n = 0 alors
  retourner ∞
tant que non x.feuille faire
  Lire-Disque(x.c_1)
  x ← x.c_1
retourner x.clé_1
```

Exercice 4 : Prédécesseur

Écrire l'algorithme qui trouve le prédécesseur d'une clé donnée d'un B-arbre.

```
B-Arbre-Prédécesseur(T, k)
x ← T.racine
pred ← −∞
tant que Vrai faire
  i ← 1
  tant que i ≤ x.n et k > x.clé_i faire
    pred ← max(pred, x.clé_i)
    i ← i + 1
  si x.feuille alors
    retourner pred
  Lire-Disque(x.c_i)
  x ← x.c_i
```

Exercice 5 : Dichotomie

On suppose que Rechercher-B-Arbre est implémentée de manière à utiliser une recherche dichotomique, plutôt qu'une recherche linéaire, à l'intérieur de chaque nœud. Montrer que le temps CPU requis devient alors $O(\log n)$, indépendamment de la façon dont t est choisi comme fonction de n.

Avec une recherche dichotomique, on réaliserait $O(\log t)$ comparaisons sur chaque nœud et il y en a $h - 1 = O(\log_t n)$ sur le chemin de recherche. On aurait donc un coût $O(\log t \log_t n) = O(\log t \frac{\log n}{\log t}) = O(\log n)$.

Exercice 6 : Variante

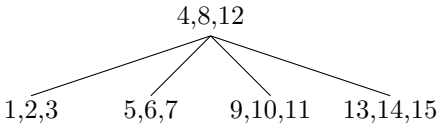
Comme les feuilles n'ont pas besoin de pointeurs d'enfant, on pourrait concevoir qu'elles utilisent une valeur de t différente (plus grande) que les nœuds internes pour la même taille de page disque. Montrer comment modifier les procédures de création et d'insertion dans un B-arbre pour gérer cette variante.

On adapterait les conditions qui vérifient si l'enfant à partager à un degré maximal. Il faudrait différencier la condition dans le cas où l'enfant est une feuille. On modifierait en tout 2 lignes car il y a 2 appels à Partager-Enfant-B-Arbre.

Exercice 7 : Hauteur minimum

Professeur Bunyan affirme que B-Arbre-Insérer produit toujours des B-arbre avec une hauteur minimale. Montrer que le professeur Bunyan se trompre en argumentant qu'avec $t = 2$ et l'ensemble de clés $\{1, 2, \ldots, 15\}$, il n'existe pas de séquence d'insertion qui produit un B-arbre avec la plus petite hauteur possible.

La plus petite hauteur est atteinte avec :



Considérons la dernière valeur insérée qui l'est forcément sur une feuille. Si c'est une insertion simple, on viendrait alors de passer par la racine et il aurait fallut la partager. Sinon, c'est une insertion qui entraîne un partage et on aurait alors eu au moins un nœud de l'arbre avec une seule clé.

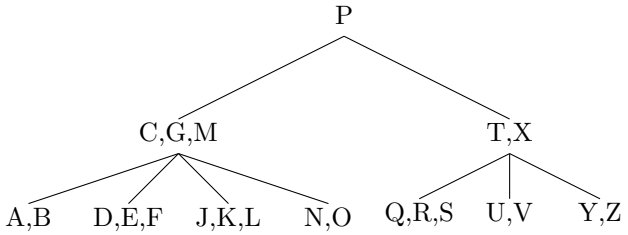
Exercice 8 : Tuning

On suppose que le disque est construit de façon qu'on puisse choisir arbitrairement la taille d'une page disque, mais que le temps pris pour lire la page est $a + bt$, où a et b sont des constantes spécifiées et t le degré minimal d'un B-arbre utilisant des pages de cette taille. Décrire la manière de choisir t pour minimiser (approximativement) le temps de recherche dans le B-arbre. Suggérer une valeur optimale de t pour le cas où $a = 5$ microsecondes et $b = 10$ microsecondes.

Si l'on prend un t trop petit, le coût sera dominé par a. S'il est trop grand, ce sera b qui dominera. Le nombre de lecture disque pour la recherche est $h - 1 \leq \log_t \frac{n+1}{2} - 1$. On cherche donc la valeur de t qui minimise $(a + bt)(\frac{n+1}{2} - 1)$. On peut dériver ou s'appuyer sur une résolution numérique (en testant toutes les valeurs).

Exercice 9 : Suppressions

Montrer le résultat de la suppression dans l'ordre de F, M, G, D, B, C, P et V dans l'arbre suivant $(t = 2)$:



Exercice 10 : Insertions

Montrer les résultats de l'insertion successive des clés F, S, Q, K, C, L, H, T, V, W, M, R, N, P, A, B, X, Y, D, Z, E dans un B-arbre vide de degré minimal $t = 2$. Ne représenter que les configurations de l'arbre qui précèdent immédiatement le découpage d'un nœud, puis représenter la configuration finale.