Mots-clés structure de données dynamique, arbres binaires de recherche, arbres équilibrés, implémentation naïve, arbres AVL, arbres rouge-noir, comparatif

Introduction

Les arbres de recherche semblent avoir été découverts indépendamment par plusieurs personnes vers la fin des années 1950. Ils permettent de gérer des ensembles dynamiques ordonnés de manière plus efficace que de simples listes chaînées. Cependant, une implémentation naïve, certes suffisante avec des données aléatoires, peut mener dans certains cas à des arbres déséquilibrés en hauteur, ce qui ôte à cette structure de données tout son intérêt.

Dans cet exposé, nous nous proposons de gérer efficacement ces cas « pathologiques » en présentant des algorithmes qui permettent la recherche et les opérations dynamiques de base — insertion et suppression — en temps logarithmique dans le pire des cas. Nous présentons les *arbres AVL* — introduits pour la première fois par Adel'son-Vel'skiĭ et Landis [1] en 1962 — et les *arbres rouge-noir* — inventés par Bayer [2] en 1972.

Performances

Implémentations Les algorithmes présentés ont été implémentés en Objective Caml. L'implémentation naïve est classique, celle des arbres AVL s'inspire très fortement de celle du module **Map** d'Objective Caml [6], et celle des arbres rouge-noir est personnelle et s'inspire de [3].

Tests effectués L'idée vient de [4]. Les tests ont été effectués sur un PC équipé d'un processeur Athlon XP 1600+ (1,4 GHz). Nous avons d'abord effectué un premier test avec des données aléatoires : les performances des trois algorithmes se sont avérées être comparables. Puis nous avons testé l'un des « cas les pires » évoqués ci-dessus en insérant n clés dans l'ordre croissant, et en relevant la hauteur de l'arbre obtenu h, et la durée de construction τ_0. Comme prévu théoriquement, les résultats (tableau 1) sont sans appel.

Bilan Les arbres AVL et les arbres rouge-noir sont donc aussi performants que l'implémentation naïve avec des données aléatoires, mais nettement meilleurs dans les pires cas. Toutefois, les arbres équilibrés stockent des données supplémentaires pour maintenir leur équilibre. Les plus performants semblent être les arbres AVL. Ce résultat était prévisible théoriquement. Ils stockent un entier supplémentaire (généralement sur 32 ou 64 bits) par nœud. Leur code est aussi plus simple. Néanmoins, les arbres rouge-noir n'ont besoin que d'un seul bit supplémentaire par nœud.

Algorithme	n	h	$\frac{h}{\log_2 n}$	τ_0 (s)	$\frac{\tau_0}{n \log_2 n}$ (μs)
Naïf	32 000	31 999	2 138	614	1 383
AVL	32 000	14	0,9	0,06	0,13
AVL	512 000	18	0,9	1,31	0,14
AVL	2 000 000	20	1,0	5,51	0,14
Rouge-Noir	32 000	26	1,7	0,08	0,17
Rouge-Noir	512 000	34	1,8	1,79	0,19
Rouge-Noir	2 000 000	38	1,8	9,76	0,24

TAB. 1 – Tests d'un des cas les pires

Perspectives

Les arbres de recherche peuvent être utiles dans des domaines aussi variés que la compilation, la gestion de fichiers ou les bases de données. Nous ne présentons ici que des arbres binaires, mais il existe aussi d'autres variantes qui agissent par des modifications du degré des nœuds. Par ailleurs, les algorithmes présentés ici ne tiennent pas compte du support de données, mais il existe aussi des algorithmes optimisés pour des arbres stockés sur disque.

Références

[1] Adel'son-Vel'skiĭ (G. M.) et Landis (E. M.). An algorithm for the organization of information. *Soviet Mathematics Doklady*, 3 : 1259–1263, 1962.

[2] Bayer (R.). Symmetric binary B-trees : Data structure and maintenance algorithms. *Acta Informatica*, 1 : 290–306, 1972.

[3] Cormen (Thomas H.), Leiserson (Charles E.), Rivest (Ronald L.) et Stein (Clifford). *Introduction à l'algorithmique*, chapitres 12–13. Dunod, 2002.

[4] Guibas (Leo J.) et Sedgewick (Robert). A dichromatic framework for balanced trees. *Proceedings of the 19th Annual Symposium on Foundations of Computer Science*, pages 8–21. IEEE Computer Society, 1978.

[5] Leroy (Xavier). *The Objective Caml system (release 3.07). Documentation and user's manual*. INRIA, 2003. Disponible sur le web (caml.inria.fr).

[6] Leroy (Xavier). *The Objective Caml system (release 3.07). The standard library*, module **Map** (fichiers **map.ml** et **map.mli**). INRIA, 2003. Disponible sur le web (caml.inria.fr).