Weekplan: B-Trees and B^ϵ-trees

Inge Li Gørtz

References and Reading

[1] External Memory Geometric Data Structures L. Arge. Lecture notes. Section 1, 2.

[2] Lower bounds for external memory dictionaries G. S. Brodal and R. Fagerberg. Section 1 and 3.4

[3] An Introduction to B^ϵ-trees and Write-Optimization. M.A. Bender, M. Farach-Colton, W. Jannen, R. Johnson, B. C. Kuszmaul, D. E. Porter, J. Yuan, and Y. Zhan.

We recommend reading [1] section 1 and 2, [2] section 1 and 3.4 in detail. [3] gives a good introduction to B^ϵ trees.

At the lecture we will talk about B-trees and a variant of these: the B^ϵ-trees (called B-trees with buffers in [2]). If time, we will also talk about Patricia tries.

Exercises

1 $[w]$ (a,b)**-trees**

1.1 Show all legal $(2,8)$-trees that represent the set $\{1,2,3,4,5\}$.

1.2 Let T be a B-tree with branching parameter b and leaf parameter $k = b$. As a function of the branching parameter b, what is the maximum number of keys that can be stored in T if T has height h?

1.3 Show the results of inserting the keys 41, 38, 31, 12, 19, 8, 43, 21, 34, 62, 1, 35, 5, 40 into an initially empty $(2,4)$-tree.

1.4 Show the trees that results from successively deleting the keys 8, 12, 19 , 31, 38, 41 in that order from the trees in the previous exercise. You should show how the trees look after each deletion.

2 Databases 1 You are working as a consultant for the company "*Boxes, Boxes and Boxes*", that sells boxes. They want a database containing information about all their boxes. Each box has an id, a size a type, and a price. They want to be able to update the database with insertions and deletions of boxes. The database should—in addition to the updates—support the following queries efficiently:

- report(a, b): Return the id of all boxes with a size between a and b.

Give a data structure supporting the required updates and queries. Analyse the space and the I/O complexity of your data structure.

3 Amortized updates in (a,b)-trees

3.1 Show that if that if $b = 2a-1$ then there exists a sequence of updates where each update results in $O(\log_a N)$ rebalancing operations.

3.2 Show that if $b = 4a$ then we can get updates in $O(\frac{\log_a N}{a})$ *amortized number of rebalancing operations* if we change the delete operation to split a newly merged node if it contains more than $3a$ elements.

Hint: Show that the number of leaf rebalancing operations performed after X operations is $O(X/a)$.

4 Secondary indices In a database, secondary indices can greatly speed up queries. Suppose you have a database with items that have a unique id k and m values, i.e., each item is a vector $(k, v_1, v_2, \ldots, v_m)$, and an application that performs queries on many different keys/values and return a function on other values in the vector, ie., it performs queries such as return $v_3 + v_8$ for all items with value v_7 in the range $[a, b]$.

To solve this you can maintain an index for each key used in a query sorted by that key. Each index has information that can be used to answer the queries. It can either have all the information required to answer the query (this is called a *covering index* for that query) or otherwise it contains keys into the primary index, which you can then use to lookup the information using a query into the primary index (see e.g. [3]).

Analyse the I/O complexity of the following queries and updates:

- a range query that is covered by the index,
- a range query that is not covered by the index,
- insertions
- deletions

if the indices are implemented using B-trees and B^ϵ trees, respectively.

5 Deletions in B-trees A deletion in a B-tree may require several I/Os in addition to those needed for locating the key. An alternative strategy would be to use tombstones to mark keys as deleted. Discuss possible disadvantages of the tombstone approach. You should consider:

- The space occupied by deleted keys.
- The time complexity of searching for a key in the B-tree.
- The time complexity of range queries.