Weekplan: External Memory II

Philip Bille

References and Reading

[1] 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, ;login: USENIX Magazine, 2015.

[2] Lower bounds for external memory dictionaries, G. S. Brodal and R. Fagerberg, SODA 2003.

We recommend reading [1] in detail. [2] is the original paper introducing B^ε-trees.

1 [*w*] **B^ε-tree Example** Solve the following exercises.

1.1 Consider a $\sqrt{B}$-tree with degree 3 and buffer size 3. Insert the sequence $S = 1, 2, 3, \ldots, 10$. At the end show the action of searching for each element in S.

1.2 Consider inserting and deleting the same element multiple times in a B^ε-tree. How will the buffers in the tree reflect the current state of the element? How can we correctly perform search?

2 [*w*] **B^ε-tree Analysis** Analyze I/O bounds for searches and updates on a B^ε-tree. What happens if we set $\varepsilon = 1$?

3 [*w*] **B^ε-tree Node Implementation** Give an efficient internal memory implementation of a B^ε-tree node. *Hint:* what operations are needed and what data structure supports these efficiently?

4 **Shared Buffers in B^ε-trees** Each node in a B^ε-tree stores B^ε separate buffers (one for each child) of size $B^{1-\varepsilon}$. A friend suggest to store all the buffers together in a single buffer of size B. We only flush the buffer if it is completely full and hence potentially avoid some of the flushing if updates are uneven. Will this modification work and how will it affect the performance of B^ε-trees?

5 **Range Searching** Suppose we want to extend B^ε-trees to support the following range searching operations:

- report(i, j): Report all elements with keys k, such that $i \leq k \leq j$.
- count(i, j): Return the number of elements with keys k, such that $i \leq k \leq j$.

Solve the following exercises.

5.1 [*w*] Recall (or reproduce) the bounds for these operations on B-trees.

5.2 Can you match these bounds with a B^ε-tree?

6 **Advanced Updates in B^ε-Trees** Consider a B^ε tree storing a set of keys S. Each key x also stores an integer x.data. Consider the following update operations:

- increment(x): If $x \in S$, add 1 to x.data. Otherwise, do nothing.
- delete-predecessor(x): If $x \in S$, delete the largest key that is $< x$.
- delete-if-negative(x): If $x \in S$ and x.data < 0 delete x. Otherwise, do nothing.

Solve the following exercises.

6.1 Consider implementing the above operations. For each of them either give an efficient buffered implementation or argue why they cannot be implemented with the buffering technique.

6.2 In general, what operations can be efficiently implemented with the buffering technique?