Weekplan: External Memory I

Philip Bille

References and Reading

[1] The Input/Output Complexity of Sorting and Related Problems, A. Aggarwal and J. Vitter, CACM 1988. Set $P = 1$ when reading this.

[2] Organization and Maintenance of Large Ordered Indexes, R. Bayer, E. McCreight, Acta Inform., 1972.

[3] Introduction to Algorithms, 3rd edition, Chap. 18, T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, 2009.

We recommend reading [1] and [3] in detail. [2] is the original paper introducing B-trees.

Exercises

1 [w] **Prefix Sum** Given an array A of N elements, the *prefix-sum* of A is the array P such that $P[i] = \sum_{j \le i} A[j]$. Show how to compute the prefix sum of A efficiently in external memory

2 [w] **Memory Hierarchy** Determine the configuration of the memory hierarchy on your own computer. Also, what is the cache-inclusion policy?

3 Stacks and Queues Consider stacks and queue in external memory. Solve the following exercises.

3.1 Show how to efficiently implement a stack in external memory. What is the worst-case and amortized I/Os per operation?

3.2 Do the same for a queue.

4 RAM algorithms in External Memory We can implement any standard RAM algorithm directly in external memory as follows:

- When we access a piece of data that is not already in internal memory, we move the block containing the input data into internal memory.
- When the internal memory is becomes full, we write the block that contains the *least recently used* (has not been used for the longest amount of time) data back to disk.

Solve the following exercises.

4.1 Consider your favourite sorting algorithm. What is the I/O complexity of this algorithm if implemented directly in external memory? Compare the result with a good external algorithm.

4.2 Consider your favourite data structure for searching. What is the I/O complexity of this algorithm if implemented directly in external memory? Compare the result with a good external data structure.

5 Multiway Merge Sort Analysis Carefully analyse the complexity of the multiway merge sort and algorithm and show that it uses $O(N/B \log_{M/B}(N/B))$ I/Os.

6 Linked Lists Consider a data structure that maintains a sequence of elements $L = e_1, \ldots, e_N$ under the following operations:

- insert(e, e'): Insert element e' immediately after element e in the sequential order in L (extending the length of the sequence by 1).
- delete(e): Delete the element e in L.
- traverse(): Report the elements in L in sequence.

We assume that the arguments e and e' are pointers to elements. Show how to efficiently implement the operations in external memory. *Hint:* What is the optimal I/O bound you can hope to achieve for the traverse operation? Try to achieve that.

7 Range Searching Suppose we want to extend B-trees to support the following range searching operations:

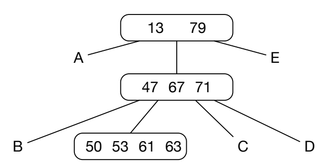
- report(i, j): Report all elements with keys k, such that $i \le k \le j$.
- count(i, j): Return the number of elements with keys k, such that $i \le k \le j$.

Solve the following exercises.

7.1 Show how to efficiently implement report. Your solution should have a good dependency on the size of the output.

7.2 Show how to efficiently implement count.

8 Insertions in B-tree Consider the following B-tree of order 4. The capital letters represent subtrees. Show the tree after inserting 59.



9 B-tree Construction Show how to efficiently construct a B-tree from an array of N elements.

10 Optimality of B-trees Suppose that we want to search among N keys. Furthermore, suppose that the only way of accessing disk blocks is by following pointers. Show that a search takes at least $\Omega(\log_B N/M)$ I/Os in the worst case. Also, compare this bound to the B-tree upper bound. *Hint:* Consider the size C_t of the set of blocks that can be accessed in at most t I/Os. Assume that our memory initially is full of pointers.

11 Dynamic Programming Let S and T be strings of length N and consider the classic $O(N^2)$ time solution for computing the longest common subsequence of S and T. Show how to implement the algorithm efficiently in external memory.