Weekplan: External Memory III

Philip Bille   Inge Li Gørtz   Eva Rotenberg

References and Reading

[1] Cache-Oblivious Algorithms and Data Structures, Erik Demaine, Lecture Notes from the EEF Summer School on Massive Data Sets, 2002

[2] Cache-Oblivious Algorithms, M. Frigo, C.E. Leiserson, H. Prokop, S. Ramachandran, FOCS 1999

We recommend reading [1] in detail. [2] is the paper introducing the cache-oblivious model.

1 [w] **Double Array Traversal** Consider arrays $A = [1,2,3,4,5,6,7,8]$ and $B = [9,10,11,12,13,14,15,16]$ and function $f(A[i], B[j]) = A[i] + B[j]$). Draw the tree of recursive subproblems in the cache-oblivious algorithm for double array traversal.

2 **String Reversal** Let S be a string of length N stored in $O(N/B)$ blocks. We want to compute the *reverse* string S^R of S. Solve the following exercises.

2.1 Give an efficient algorithm to reverse S in the I/O model.

2.2 Give an efficient algorithm to reverse S in the cache-oblivious model.

3 **Stacks and Queues** Show how to efficiently implement stacks and queues in the cache-oblivious model. Can you match the cache-conscious I/O bounds?

4 **Cache-Oblivious Analysis** Solve the following exercises.

4.1 Analyse binary search in the cache-oblivious model. What is the dependency on B?

4.2 Analyse mergesort in the cache-oblivious model.

5 [w] **van Emde Boas Layout** Consider a complete binary tree T of height 3 with 15 nodes. Solve the following exercises.

5.1 Draw T and number each node with it positions in the vEB layout.

5.2 Draw two new copies of T and number the nodes according to the *heap layout* (layout used in binary heaps) and the *inorder layout* (ordering corresponding to the inorder traversal of T). Compare these with the vEB layout.

6 **van Emde Boas Ordering** The vEB layout orders the recursive layout of the bottom trees from left-to-right. Suppose we reverse this ordering. How does this change the performance of the layout?

7 Cache-Oblivious Lookahead Array Consider the following dynamic search data structure called the *cache-oblivious lookahead array* (COLA). It consists of $\lceil \log_2 N \rceil$ arrays each of which is either completely full or completely empty. The kth array is of length 2^k and contains items iff kth least significant bit of N is 1. Each of the full arrays stores items in sorted order. Solve the following exercises.

- **7.1** [w] Draw a small example of a COLA contains 9 items.
- **7.2** Show how to search a COLA in $O(\log^2 N)$ I/Os.
- **7.3** Show how to insert elements into a COLA in $O(\log N)/B$ amortized I/Os. *Hint:* think binary addition and merging.
- **7.4** [$*$] Show how to search a COLA in $O(\log N)$ I/Os. *Hint:* fractional cascading.

8 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 the cache-oblivious model (if you have not done the earlier exercise on dynamic programming in the I/O model, do so before this exercise).