Weekplan: External Memory

Philip Bille

References and Reading

[1] Cache-oblivious dynamic programming, R. A. Chowdhury and V. Ramachandran, SODA 2006.

[2] The Input/Output Complexity of Sorting and Related Problems, A. Aggarwal and J. Vitter, CACM 1988

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

We recommend reading [1] and [3] in detail. [3] define the I/O model and cache-oblivious model and covers most basic cache-oblivious algorithms. [1] presents the cache-oblivious algorithm covered in the lecture. [2] is the original paper defining the I/O model.

Exercises

1 Shortest Paths in Implicit Grid Graphs

1.1 [w] Compute the edit distance between strings `survey` and `surges`. Fill in the dynamic programming table.

1.2 [w] Let S and T be strings of length n. Show how to compute the length of the longest common subsequence of S and T in $O(n^2)$ time and $O(n)$ space.

2 External Memory Shortest Paths in Implicit Grid Graphs Consider the 7×7 dynamic programming matrix for `survey` and `surges`. Partition the matrix into overlapping 4×4 submatrices and explicitly write the information needed to compute each submatrix. What is the minimum value of M needed for an I/O efficient submatrix computation?

3 Dynamic Programming meets Divide and Conquer Consider the dynamic programming algorithm for the shortest path in implicit graphs problem in the RAM model. We are interested in efficiently computing not only the length of the shortest path but also the edges on the shortest path. Solve the following exercises.

3.1 Show that with $O(n^2)$ space we can compute the path in $O(n^2)$ time.

3.2 Show that we can compute a single edge on the shortest path corresponding to the $n/2$th row in the graph using $O(n)$ space and $O(n^2)$ time.

3.3 Show how to recursively apply 2 to output the shortest path in $O(n)$ space and $O(n^2)$ time.

4 Stacks and Queues in External Memory Show how to implement stacks and queues with $O(1/B)$ amortized I/Os per operation in the I/O model of computation.

5 External Sorting We want to sort an array of N numbers in the I/O model efficiently. Solve the following exercises.

5.1 Show how to merge $\Theta(M/B)$ sorted arrays of total length N into a single sorted array in $O(N/B)$ I/Os.

5.2 Given an unsorted array of length N, show how to create $\Theta(N/M)$ sorted arrays of each of length M in $O(N/B)$ I/Os.

5.3 Show how to sort an array of length N using

$$O\left(\frac{N}{B}\log_{M/B}\frac{N}{M}\right)$$

I/Os. *Hint:* Do a multiway merge using 1 for merging and 2 as base case.

6 Tree Layout in External Memory Let T be a complete binary tree with $N = 2^h - 1$ nodes. The leaves of T stores a set S of numbers sorted in increasing order from left-to-right in T. Each internal node in T stores the maximum and minimum number stored in its descendant leaves. A *top-down search* for a number x traverses T from the root to a leaf ℓ and returns ℓ if ℓ stores x and otherwise reports that x is not in S. A *layout* of T maps each node in T to a location on disk. We want to design layouts of T that supports I/O efficient top-down searches of T. Solve the following exercises.

6.1 Suppose we layout T according to an inorder traversal of T. Specifically, we store T in an array A of length N using $O(N/B)$ blocks. The root is stored in the center of A and the left and right subtrees of T are stored recursively in the left and right half of A. Analyse the number of I/Os needed for a top-down search of T in the I/O model.

6.2 Show how to layout T efficiently in the I/O model. The number of I/Os should be asymptotically smaller than the previous exercise. *Hint:* Partition the tree.

6.3 Suppose we layout T according to the following recursive layout.

- If $N = O(1)$, layout the nodes in T according to a inorder traversal of T in an array of size N.
- Otherwise, partition T into a *top tree* T_{top} consisting of all nodes of depth at most $h/2$ and a number of *bottom trees* $T_1, \ldots, T_k$ defined as the connected subtrees obtained by removing the top tree. Recursive layout T_{top} in an array A_{top}, and $T_1, \ldots, T_k$ in arrays $A_1, \ldots, A_k$, respectively. The layout for T is the array $A_{\text{top}} \cdot A_1 \cdot A_2 \cdots A_k$, where $\cdot$ denotes concatenation.

Analyse the number of I/Os needed for a top-down search of T with this layout in the cache-oblivious model.

7 Parallel Dynamic Programming Consider the standard dynamic programming algorithm for the shortest path in implicit graphs problem. Suppose that we have $p > 1$ processors at our disposal. How can we use these to speedup the standard dynamic programming solution in the RAM model? What about the I/O model or cache-oblivious model?

8 Medians Let A be an array of N numbers. Show how to find the median of A in $O(N/B)$ I/Os. *Hint:* The classical divide and conquer linear time RAM algorithms works.