Mandatory Exercise: External Memory

Philip Bille

1 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.

1.1 Suppose we layout T according to an inorder traversal of T. Specifically, we store T in an array A of length N using $\lceil N/B \rceil$ 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.

1.2 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.

1.3 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.