6.851 Advanced Data Structures (Spring'12)

Prof. Erik Demaine TAs: Tom Morgan, Justin Zhang

$\boxed{\text{Problem 4}}$ *Sample solution*

Cache-oblivious median finding. The standard median of medians algorithm is already a cache-oblivious algorithm meeting the desired bounds, so all the remains is to prove this. Each step of the recursion (finding the medians of the groups of five and selecting the elements for recursion) can be done in $O(1)$ passes over the the array for a total of $O(\lceil N/B \rceil)$ memory transfers. The number of memory transfers is then

$$T(N) = T(N/5) + T(7N/10) + O(\lceil N/B \rceil).$$

With a base case of $T(B) = O(1)$ since once the list fits in a single line of cache, the median can be found with a single memory transfer. $T(N)$ decreases geometrically from the root, down to $O((N/B)^c)$ $(c < 1)$ work at the leaves. Thus, $T(N) = O(\lceil N/B \rceil)$.

Cache-oblivious queue. We will store the items in the queue sequentially in external memory from an index *head* to an index *tail*. Initially, $head = tail = 0$. To enqueue an item, we store it at index *tail*, and we add 1 to *tail*. To dequeue an item, we report the item stored at index *head* and add 1 *head*. Assuming optimal cache behavior, and $M/B \geq 2$, enqueue and dequeue clearly take an amortized $O(1/B)$ memory transfers per operation. This is because we can just keep the lines containing *head* and *tail* in cache at all times so a memory transfer only occurs one out of every B enqueues or dequeues.

Note that this algorithm does not stay in memory indices $\{0, 1, \cdots, O(N)\}$. To fix this, whenever $2 * head > tail$, we shift all the elements to the beginning of external memory. This shift takes $O(k/B)$ memory transfers, where k is the current size of the queue, and we can charge it to the $\Omega(k)$ dequeues that must have occurred between it and the previous shift, so we still have an amortized $O(1/B)$ memory transfers per operation.