Problem session

Advanced database technology

February 14, 2005

This problem considers an external memory analog of a linked list in external memory. The essential properties of a (doubly) linked list are:

- We can have a pointer into any position of the linked list.
- The pointer can be moved to the next or previous position in constant time.
- New items can be inserted next to the pointer position in constant time.
- The item pointed to can be deleted in constant time. (This moves the pointer to an adjacent item.)

In external memory a block can hold B items. The obvious way of implementing a doubly linked list of N elements is to keep the elements in $\lceil N/B \rceil$ blocks linked together in a doubly linked list, each block, except possibly the last, holding B elements. (We disregard the space needed for pointers, assuming that a block holds up to B elements plus pointers.)

1. How many I/Os are needed in the worst case to insert or delete an element if we use the obvious implementation?
2. We now investigate the possibility of allowing free space in the blocks when deleting elements. Obviously this makes deletions much more efficient. Show that it also allows insertion in $O(1)$ I/Os. What is the problem with this solution?
3. We now consider the following invariant which allows blocks to be partially, but not too, empty: *In any block there is at least $B/3$ elements.*
   (a) Show that the invariant implies that the entire list can be traversed in at most $3N/B$ I/Os, i.e., $3/B$ I/Os per element.
   (b) Show how to maintain the invariant when we try to place an element in a block already holding B elements.
   (c) Similarly, show how to maintain the invariant when an element is deleted. That is, if we delete an element such that the block of the element is too empty, show that the invariant can be reestablished efficiently. (You have to consider different cases depending on the number of elements in the neighboring blocks.)

Extra problem for students liking a challenge: Suppose that along with a pointer to an element we always store (in internal memory) the two most recently visited blocks. Show that all the constant time operations from the internal memory linked list can, in this way, be implemented in $O(1/B)$ I/Os, amortized.