**Operations**:-

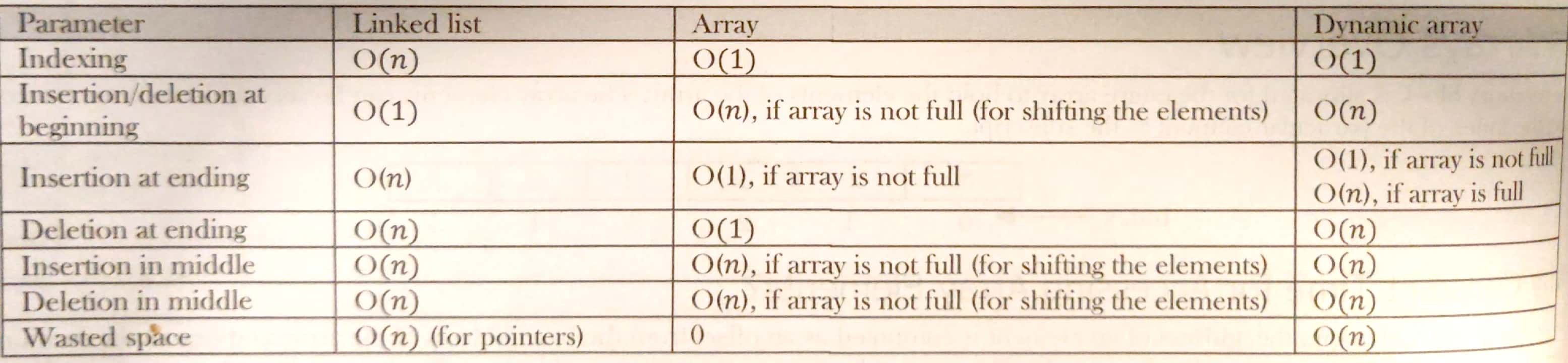
* Insert : inserts an element into the list
* Delete: removes and returns the specified position element from the list
* Delete List: Removes all elements of the list
* Count: Returns the number of elements in the list
* Find nth node from the end of the list

**Advantages**:-

* Expanded in constant time

**Disadvantages**:-

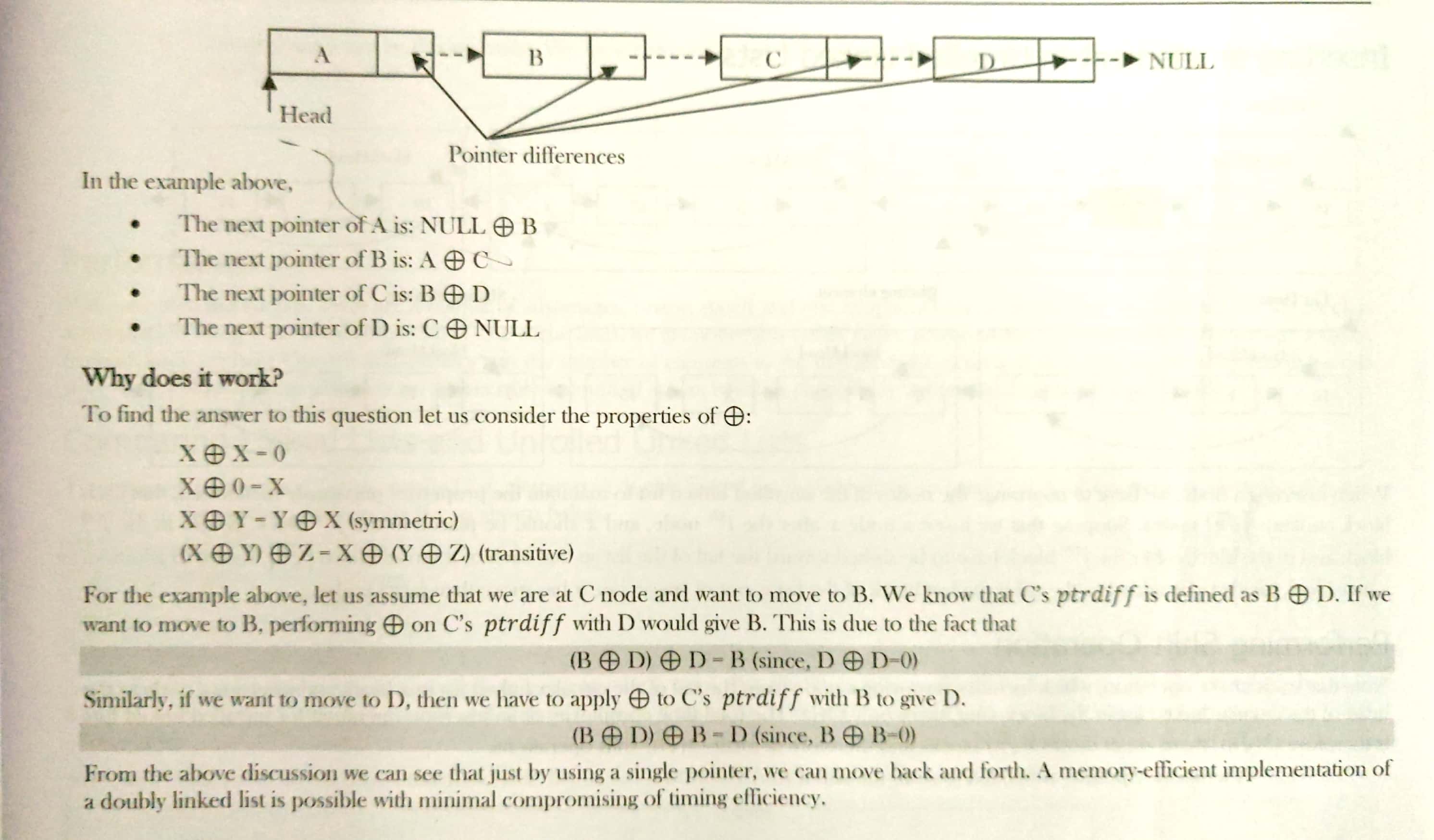
* Linked lists take O(n) for access to an element in the list in the worst case
* Linked lists waste memory in terms of extra reference points.



**Singly Linked List**

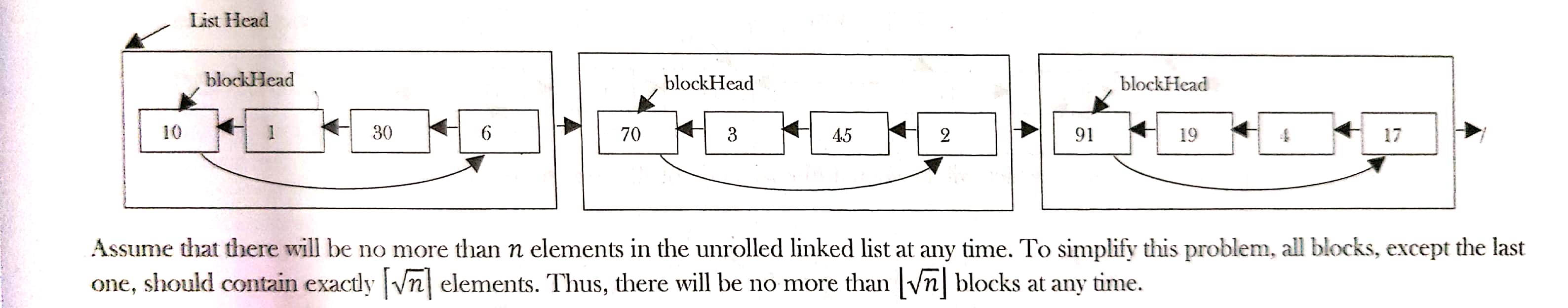
The first part of the record is a field that stores the data, and the second part of the record is a field that stores a pointer to a node.

Each node is allocated in the heap with a call to malloc(), so the node memory continues to exist until it is explicitly deallocated with a call to free().

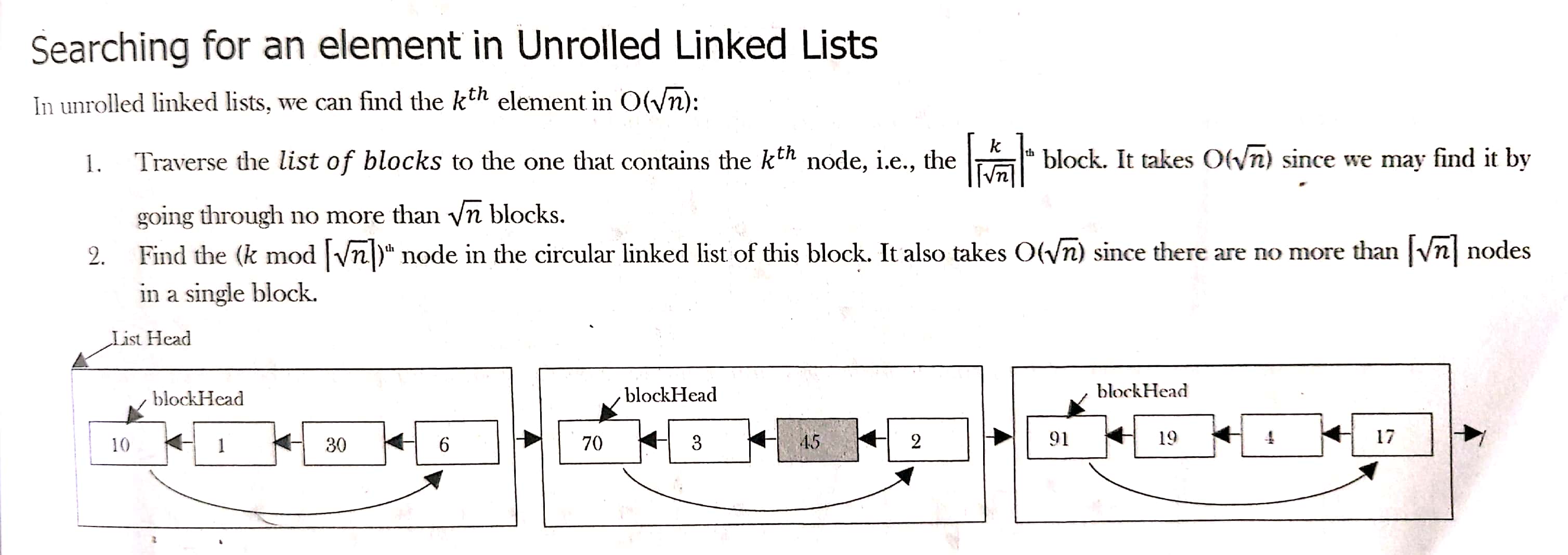


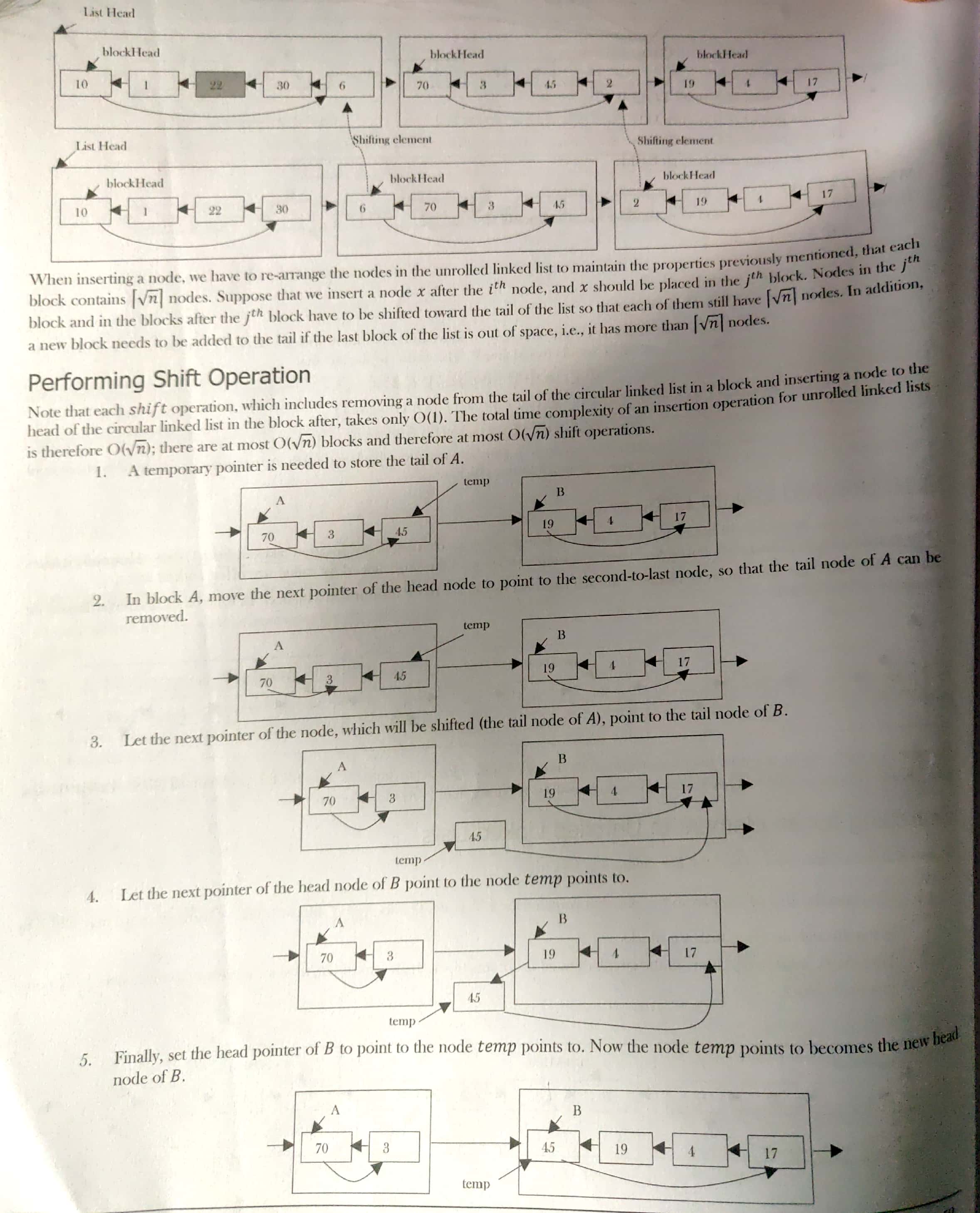
**Unrolled Linked List**

An unrolled linked list stores multiple element in each node (let us call it a block for our convenience). In each block, a circular linked list is used to connect all nodes.



**Searching for an element in Unrolled Linked List**





6. temp pointer can be thrown away. We have completed the shift operation to move the original tail node of A to become the new head node of B.

**Performance**

With unrolled linked lists, there are a couple of advantages, one in speed and one in space. First, if the number of elements in each block is appropriately sized (at most the size of one cache line), we get noticeably better cache performance from the improved memory locality. Second, since we have O(n/m) links, where n is number of elements in the unrolled linked list and m is the number of elements we can store in any block, we can also save an appreciable amount of space, which is particularly noticeable if each element is small.