Shopping Cart Report using LinkedList

Built a Shopping Cart system using a linkedlist where each node stores a data and a pointer that points to another node. In my program I input clothing items such as Nike’s clothing items, Adidas items, and New Balance items. Along with duplicate items in the list. By implementing a linkedlist, I did not have to request for memory capacity space or about resizing memory. In this report, I discuss the advantages and disadvantages of my approach and ways I could have improved it.

**The Pros**

By implementing a linkedlist which is a linear data structure that uses nodes to point to another node. We are able insert nodes without having to provide a contiguous block of memory. In addition, when it comes to inserting and removing nodes in the middle of a linear data structure, there is less overhead because we do not have to reallocate memory.

When it comes to the time complexity of Linked Lists

|  |  |
| --- | --- |
| Operation | Time Complexity |
| Search | O(n) |
| Insert/Delete | 1. O(1) if in the beginning 2. O(n) because you have to traverse the list. **This can be improved to O(1) if a last node pointer is created and updated.** 3. O(n) in the middle |
|  |  |

**The Cons**

The cons with implementing a linkedlist is when searching through the list of nodes. Because when searching for an item, we have to start at the head of the node and traverse through the list, checking each node to see if it has the data or value we are looking for. There is no random access like with an array.

Another possible con with implementing a linkedlist is if you don’t implement a last node pointer. This is because adding an element in a linkedlist becomes O(n). You would need to traverse the list to find the last node.

However, this can be solved by creating a last node pointer and updating the last node pointer as new nodes get added. This makes it so that adding a new node becomes O(1).