Isaiah Lugo

CSM III – Data Structures

Final Project: Part 1

Algorithm 1: Find Element in Sorted Array with Duplicates

* Data Structure: Array with Binary Search.
* Why? Since the array is sorted, binary search is optimal for finding elements efficiently in O(log n) time. To find the first occurrence of a duplicate, we can modify binary search to keep moving left whenever the target is found, ensuring the leftmost index is returned.

Algorithm 2: rotate\_left\_until\_zero()

* Data Structure: Array with Rotation Operation.
* Why? Using the array directly allows rotation without additional space. A simple traversal to locate the zero, followed by a slicing operation, rearranges the elements with minimal time complexity. This keeps the algorithm efficient in O(n), where n is the length of the array.

Algorithm 3: Reverse Polish Notation

* Data Structure: Stack.
* Why? A stack is ideal for evaluating RPN expressions because it allows efficient management of operands and operators. Pushing operands to the stack and performing operations when operators are encountered achieves the solution in O(n), as each element is processed once.

Algorithm 4: Merging Linked Lists

* Data Structure: Linked List.
* Why? Linked lists allow efficient splicing and rearranging of nodes, making them suitable for merging two sorted lists. By iterating through both lists, we can merge them in O(n+m) time, where n and m are the lengths of the two lists, maintaining the sorted order without extra memory allocation.