Documentetion

Intuition

The problem involves parsing a serialized string representation of nested integers and lists into a structured object called NestedInteger. The structure supports two types of elements: single integers and nested lists. The challenge lies in identifying the correct hierarchy of lists and integers based on the string format and ensuring the nested relationships are preserved during parsing.

Overview

The string input can represent either a single integer or a more complex structure with nested brackets ([and]). To handle this, the approach needs to differentiate between single integers and nested lists. When brackets are encountered, they indicate the start or end of a nested list, while commas separate elements. Numbers (including negatives) need to be parsed and added to the correct NestedInteger object.

Approach

The solution uses a stack to manage nested structures. When encountering an opening bracket [, a new NestedInteger object is created, and the previous one (if any) is pushed onto the stack for later reference. When a closing bracket] is encountered, the current NestedInteger is finalized, and if there is a parent in the stack, the finalized object is added. Numbers are parsed as they appear in the string and are added to the appropriate NestedInteger.

Handling Edge Cases

If the input string is a single integer (no brackets), the solution directly initializes and returns a NestedInteger containing that integer. For nested lists, careful attention is given to managing incomplete numbers (those still being parsed) and ensuring they are added to the current NestedInteger when necessary. Additional edge cases include handling negative numbers and empty nested lists.

Time Complexity

The algorithm processes each character in the input string exactly once. Operations such as appending elements to a list, pushing/popping from a stack, and adding integers are performed constantly. Thus, the overall time complexity is **O(n)**, where n is the length of the input string.

Space Complexity

The space complexity is determined by the depth of the nested structure and the stack size used to manage it. In the worst case, where the input represents a deeply nested list, the stack size grows linearly with the nesting depth. Hence, the space complexity is **O(n)**.

Key Takeaways

This problem highlights the importance of parsing and hierarchical data management. The algorithm maintains proper parent-child relationships between nested lists by leveraging a stack. The approach also demonstrates how simple operations like character traversal can be combined with logical structures (like stacks) to handle complex hierarchical data efficiently.