DAY 12:

Task 6:

Q) Searching for a Sequence in a Stack

Given a stack and a smaller array representing a sequence, write a function that determines if the sequence is present in the stack. Consider the sequence present if, upon popping the elements, all elements of the array appear consecutively in the stack.

EXPLANATION:

Steps Explained

1. Reverse the Sequence:

- The sequence array is reversed to facilitate checking from the bottom to the top of the stack. This allows us to compare the elements in the order they appear when accessed from the bottom of the stack.

2. Push Sequence into a Temporary Stack:

- A temporary stack is used to hold the reversed sequence. This allows us to compare elements easily using stack operations.

3. Comparison Process:

- Iterate through the original stack and compare each popped element with the top element of the temporary stack.

- If the popped element matches the top of the temporary stack, pop the top of the temporary stack (indicating that part of the sequence has been found).

4. Final Check:

- After iterating through the original stack, check if the temporary stack is empty.

- If the temporary stack is empty, it means all elements of the sequence have been matched in the correct order.

Example Workflow

Consider a stack with elements [8, 16, 12, 25, 5] (top to bottom) and a sequence [5, 25, 12]:

1. Reversing the Sequence:

- The sequence [5, 25, 12] is reversed to [12, 25, 5].

2. Temporary Stack Initialization:

- Push the reversed sequence into a temporary stack: [12, 25, 5].

3. Comparison Process:

- Pop 5 from the stack and compare it with 12 (top of the temporary stack). No match, continue.
- Pop 25 from the stack and compare it with 12. No match, continue.
- Pop 12 from the stack and compare it with 12. Match found, pop 12 from the temporary stack.
- Continue this process until the temporary stack is empty or the original stack is exhausted.

4. Final Check:

- If the temporary stack is empty, the sequence [5, 25, 12] is found in the stack in the correct order.

Conclusion

The provided algorithm efficiently checks whether a given sequence of integers is present in a stack by using a temporary stack to manage the sequence and performing element comparisons through stack operations. This approach ensures that the sequence is identified correctly while maintaining the order from bottom to top of the stack.