Example - The POP(i) Stack Operation

Idea:

- Although pointer based list implementations usually save time by allowing a larger range of operations that can be done efficiently, sometimes there are "cheap" tricks that can be done with array implementations.
- Fortunately, in some cases, the total time of the corresponding work done in the pointer implementation for a sequence of operations is not so bad.
- The POP(i) operation is an example of this.

**Definition:** The operation POP(i) removes $i-1$ items from the stack and then does a normal POP operation (if the stack has less than $i$ items, there is an underflow error).
Problem: With an array implementation, in $O(1)$ time we can simply subtract $i$ from $top$ and return $stack[top]$, but with pointers we must do:

```plaintext
procedure POP(i):
    for $j := 1$ to $i-1$ do POP
return POP
end
```

Note: We use POP as both a procedure that discards the element on top of the stack and as a function that deletes and returns it.

Idea: A POP($i$) can be $\Omega(n)$ time. However, $n$ operations are $O(n)$ because the cost of a POP($i$) can be charged against the PUSH operations that placed the $i$ elements.
Theorem: Starting with an empty stack, a sequence of \( n \) PUSH and POP(\( i \)) operations takes \( O(n) \) time.

Proof: Give each PUSH two "credits" and each POP(\( i \)) one credit, where a credit "pays" for \( O(1) \) computing time.

- When a PUSH is performed, one credit pays for the operation and the other credit is associated with the item.
- When a POP(\( i \)) is performed, the credit associated with each item that is popped can be used to pay for that iteration of the \textit{while} loop, and the credit with the POP(\( i \)) pays for the remaining \( O(1) \) time for the operation.
- Hence, the total credits used (which is proportional to total time used) is linear in the number of operations.
Amortized Time Analysis

This analysis of the time used by the POP\((i)\) operation is a simple example of an *amortized* time bound.

To show that a sequence of \(n\) PUSH and POP\((i)\) operations are \(O(n)\), we did not show that each POP\((i)\) operation was \(O(1)\) in the worst case.

In fact, an individual POP\((i)\) can be \(\Omega(n)\).