Amortize push pop

Below is an explanation of the note on **amortized analysis** for push, pop, and multipop operations on a stack:

**Problem Setting**

We have a stack supporting three operations:

1. **push(x)**: Pushes an element xx onto the stack.
2. **pop()**: Pops (removes) the top element from the stack.
3. **multipop(k)**: Pops the top kk elements from the stack (or until the stack is empty if it has fewer than kk elements).

We want to analyze the **total cost** of a sequence of nn such operations on an initially empty stack and then derive the **amortized cost** per operation.

**Aggregate Analysis**

1. **Push**: Each push adds one new element to the top of the stack.
2. **Pop / Multipop**: Removing an element (whether by pop or multipop) can happen at most once for each time the element was pushed. In other words, an element cannot be removed more times than it was inserted.

**Key Observation**  
Even if a **multipop(k)** appears expensive in isolation (it might remove up to kk elements at once), each of those elements must have been **pushed** previously. Since an element can be popped only once after being pushed, the total number of actual “pop” actions (individual element removals) across the entire sequence of nn operations is bounded by the total number of pushes.

Hence:

* **Total pushes** in the sequence ≤n\leq n.
* **Total pops (including those in multipop)** ≤\leq total pushes ≤n\leq n.

Thus, over nn operations, **all** push/pop actions together account for **O(n)** total element insertions and removals.

**Amortized Cost**

Since the **aggregate** (total) cost of these nn operations is O(n)O(n), the **amortized cost** per operation is:

O(n)n=O(1).\frac{O(n)}{n} = O(1).

This means that, on average, each operation (including multipop) costs constant time, even though a single multipop could remove many elements at once.

**Summary**

1. **Push**: Costs O(1)O(1) per element.
2. **Pop** or **multipop**: Each element is popped at most once per push, so total popping across all operations is at most O(n)O(n).
3. **Amortized Analysis**:
   * Total cost of nn operations is O(n)O(n).
   * **Amortized cost** per operation is O(1)O(1).

Therefore, **although a single multipop can seem expensive**, in the grand total of all operations, **each element is only removed once**, leading to an **amortized constant time** O(1)O(1) for each operation.