CSC263 Tutorial #6 Exercises

Amortized Analysis

Alexander He Meng

Prepared for February 14, 2025

Question #1: Correctness of Queue Implementation

First, convince yourself and your group that this is a correct implementation of the Queue ADT. Trace some examples.

Proof.
To verify the correctness of the implementation, let's trace the operations step by step:

- **Enqueue Operations**:
  - When an element is enqueued, it is pushed onto stack $S1$.
  - If $S1$ has 12 or more elements and $S2$ is empty, all elements from $S1$ are popped and pushed onto $S2$. This ensures that the oldest elements are moved to $S2$, maintaining the FIFO order.
- **Dequeue Operations**:
  - When an element is dequeued, it is popped from $S2$.
  - If $S2$ is empty, all elements from $S1$ are moved to $S2$ before performing the pop operation. This ensures that the oldest element is always at the top of $S2$.

Example:

- Enqueue $1, 2, 3$: $S1 = [1, 2, 3]$, $S2 = []$.
- Dequeue: Move $1, 2, 3$ to $S2$, then pop 1. $S1 = []$, $S2 = [3, 2]$.
- Enqueue 4: $S1 = [4]$, $S2 = [3, 2]$.
- Dequeue: Pop 2. $S1 = [4]$, $S2 = [3]$.

This confirms that the implementation correctly simulates a queue. □

Question #2: Aggregate Method for Amortized Cost

Consider the sequence of operations that consists of 50 Enqueue operations followed by 50 Dequeue operations. Use the aggregate method to compute the amortized cost per operation for this sequence.

Proof.

- **Enqueue Operations**:
  - Each Enqueue operation involves pushing an element onto $S1$, which costs 2 units of time.
  - If $S1$ reaches 12 elements and $S2$ is empty, all 12 elements are moved to $S2$. This involves 12 pops (3 units each) and 12 pushes (2 units each), totaling $12 \times 3 + 12 \times 2 = 60$ units of time.
  - For 50 Enqueue operations, the worst-case cost occurs when $S1$ is full every 12 operations. This happens $\lfloor 50/12 \rfloor = 4$ times, with a total cost of $4 \times 60 = 240$ units.
  - The remaining $50 - (4 \times 12) = 2$ Enqueue operations cost $2 \times 2 = 4$ units.
  - Total cost for Enqueue operations: $240 + 4 = 244$ units.
- **Dequeue Operations**:
  - Each Dequeue operation involves popping an element from $S2$, which costs 3 units of time.
  - If $S2$ is empty, all elements from $S1$ are moved to $S2$. For 50 Dequeue operations, this happens once at the beginning, costing 60 units (as calculated above).
  - Total cost for Dequeue operations: $50 \times 3 + 60 = 210$ units.
- **Total Cost**:
  - Total cost for 100 operations: $244 + 210 = 454$ units.
  - Amortized cost per operation: $454/100 = 4.54$ units.

Thus, the amortized cost per operation is 4.54 units. □

Question #3: Accounting Method for Amortized Cost

Now consider any sequence of m Enqueue and Dequeue operations. Use the accounting method to derive an upper-bound on the amortized cost per operation.

Proof.

- **Assign Costs**:
  - Assign an amortized cost of 5 units to each Enqueue operation and 0 units to each Dequeue operation.
  - This ensures that each Enqueue operation pays for its own push (2 units) and contributes 3 units to cover future pops and moves.
- **Invariant**:
  - The 3 units contributed by each Enqueue operation are stored as "credit" to cover the cost of moving elements from $S1$ to $S2$ during Dequeue operations.
- **Cost Analysis**:
  - Each Enqueue operation costs 2 units, leaving 3 units as credit.
  - Each Dequeue operation costs 3 units, which is covered by the credit from previous Enqueue operations.
  - If $S1$ is full and $S2$ is empty, the cost of moving 12 elements is 60 units, which is covered by the $3 \times 12 = 36$ units of credit from the 12 Enqueue operations.
- **Upper Bound**:
  - The total amortized cost for m operations is $5m$ units.
  - Thus, the amortized cost per operation is at most 5 units.

Therefore, the amortized cost per operation is $O(1)$, with an upper bound of 5 units. □