

Prob. 1	Prob. 2	Prob. 3

Problem 1.

Stack is capable of inserting and deleting at the end of a list but queue accepts elements from one end and elements are removed from the other end. However, a queue can be implemented by two stacks, namely *inbox*, S_i and *outbox*, S_o .

We will use S_i for Insert-Queue and S_o for Delete-Queue operation. Therefore, we push an element into S_i to insert a new element. Delete operation contains two scenarios according to state of S_o . If it isn't empty, we pop its top element; otherwise, we transfer all elements from S_i to S_o respectively by popping from S_i and pushing into S_o . Then, we have the required element at the top of S_o so we pop it.

For analysis, we define potential function: $\phi = 2n_i$, where n_i is the number of elements in stack S_i . Then, we have the following amortized cost for each Insert-Queue operation:

$$actualcost + \Delta\Phi = 1 + 2 \leq O(1)$$

For each Delete-Queue operation, we have to consider two cases. When S_o is not empty, the amortized cost is:

$$actualcost + \Delta\Phi = 1 + 0 \leq O(1)$$

When S_o is empty, the amortized cost is:

$$actualcost + \Delta\Phi = 2n_i + 1 + (0 - 2n_i) \leq O(1)$$

As a result, amortized cost of each Insert-Queue and Delete-Queue operations is constant.

Problem 2.

Problem 3.