COS 226. **Bob's Notes 1: Queues and Deques as Stacks** October 5,2019
Fall 2019

Problem 1. Implement a queue using a constant number of stacks so that m queue operations starting with an empty queue take cm stack operations for some constant c.

Solution. Two stacks, F(*front*) and B(*back*), suffice. To inject x into the queue, push it on B. To pop from the queue, if F is non-empty, pop from F. If both F and B are empty, throw an exception. If F is empty but B is not, successively pop the items on B and push them on F; once B is empty, pop the top item on F and return it. from B.

Correctness. The algorithm maintains the invariant that the order of items on the queue is the same as the order of the items on F top to bottom followed by the items on B bottom to top. The algorithm maintains this invariant when it pops an item from F (the first item on the queue) or pushes an item on B (the last item on the queue). When F is empty, successively popping the items on B and pushing them on F preserves the invariant, since the items end up on F in the reverse of their order on B.

Efficiency. We prove that m_1 inject and m_2 intermixed pop operations on the queue take at most $3m_1 + m_2$ pushes and pops on the stacks. To do this we allocate credits to each queue operation. Each credit can pay for one push or pop operation. Specifically, we allocate 3 credits to each inject and 1 credit to each pop. We prove that the algorithm maintains the *credit invariant* that $|B|$ credits are unspent. The invariant gives the bound.

An inject gets three credits. One is spent to push the item on B. This increases $|B|$ by one. The additional two credits allocated to the inject are unspent and preserve the credit invariant.

A queue pop gets one credit. If F is non-empty, this credit pays for the stack pop that executes the queue pop. More interesting is what happens when F is empty. In this case the $2|B|$ unspent credits satisfying the credit invariant pay for popping each item on B and pushing it on F. Once this is done, B is empty, so the credit invariant requires no saved credits and still holds. The credit allocated to the queue pop now pays for the pop of the returned item from F.

Remark. Instead of pushing the last item popped from B onto F, we could merely return it. This saves two stack operations each time B is emptied: the total number of stack operations becomes $3m_1 + m_2 - 2e$, where e is the number of times B is emptied.

Remark. In this simple example, we could use the more local credit invariant that each item on B has two unspent credits, used to pay for moving it to F.

Problem 2. Implement a deque using a constant number of stacks so that m queue operations starting with an empty queue take cm stack operations for some constant c.

Solution. Three stacks, F, B, and A, suffice. The order of the items on the queue is the same as in the solution to Problem 1: the items on F top to bottom followed by those on B bottom to top.

We use the auxiliary stack A to re-establish the correct order of items on F and B when F is empty and a pop is done, or B is empty and an eject is done.

To push an item on the deque, push it on F; to inject an item into the deque, push it on B. To pop an item from the deque, pop it from F unless F is empty. If F and B are empty, throw an exception. If F is empty but B is not, let k be the number of items on B. Successively pop $\lfloor k/2 \rfloor$ items on B and push them on A. Successively pop all of the remaining items on B and push them on F. Successively pop each item on A and push it back on B. Now the items on F and B are in the correct order, but F is non-empty. Pop F and return the popped item. An eject is symmetric to a pop.

Correctness. Each operation maintains the correct order of F and B.

Efficiency. The analysis is a combination of the analysis in the solution to Problem 1 and that in the analysis of array resizing. We allocate enough credits to each deque operation to maintain a number of saved credits sufficient to pay for the shuffling that takes place when F is empty and a pop is done, or B is empty and an eject is done. If F is empty and B contains k elements, the shuffling takes $4\lfloor k/2 \rfloor + 2\lceil k/2 \rceil \leq 3k$ stack operations. After the shuffling, F and B are the same size (to within one). We use the following credit invariant: the number of saved credits is at least $3||F| - |B||$. Allocation of four credits to each deque operation suffices to maintain the credit invariant. (Exercise: verify this.). We conclude that the total number of stack operations to execute m deque operations is at most $4m$.