CPS 130 Homework 17 - Solution

1. In this problem we consider two stacks A and B manipulated using the following operations (n denotes the size of A and m the size of B):

   - $PushA(x)$: Push element x on stack A.
   - $PushB(x)$: Push element x on stack B.
   - $MultiPopA(k)$: Pop $\min\{k, n\}$ elements from A.
   - $MultiPopB(k)$: Pop $\min\{k, m\}$ elements from B.
   - $Transfer(k)$: Repeatedly pop an element from A and push it on B, until either k elements have been moved or A is empty.

   Assume that A and B are implemented using doubly-linked lists such that *PushA* and *PushB*, as well as a single pop from A or B, can be performed in $O(1)$ time worst-case.

   (a) What is the worst-case running time of the operations *MultiPopA*, *MultiPopB* and *Transfer*?

   Solution: When both operations have to pop the entire stack, the running time of each op is the size of the stack, so worst case *MultiPopA* runs in $O(n)$ and *MultiPopB* runs in $O(m)$ time. *Transfer* involves popping elements off of A and pushing them onto B. Since in the worst case we transfer the entire stack on A, we use n pops and n pushes for a worst case running time of $O(n)$.

   (b) Define a potential function $\Phi(n, m)$ and use it to prove that the operations have amortized running time $O(1)$.

   Solution: Define $\Phi(n, m) = 3n + m$. Initially the potential is zero, and for non-empty stacks, the potential is always positive. The amortized costs are as follows:

   PushA

   $$\begin{aligned} \hat{c}_i &= c_i + \Phi(D_{i+1}) - \Phi(D_i) \\ &= 1 + 3(n+1) + m - (3n+m) \\ &= 4 \end{aligned}$$

   PushB

   $$\begin{aligned} \hat{c}_i &= 1 + 3n + (m+1) - (3n+m) \\ &= 2 \end{aligned}$$

   MultiPopA

   $$\begin{aligned} \hat{c}_i &= k + 3(n-k) + m - (3n+m) \\ &= -2k \end{aligned}$$

   MultiPopB

   $$\begin{aligned} \hat{c}_i &= k + 3n + (m-k) - (3n+m) \\ &= 0 \end{aligned}$$

   Transfer

   $$\begin{aligned} \hat{c}_i &= 2k + 3(n-k) + (m+k) - (3n+m) \\ &= 0 \end{aligned}$$

The amortized cost of each function is bounded above by a constant, so the overall run time for all operations is $O(1)$. It is ok to have a negative amortized cost in the *MultiPopA* example. *PushA* pays for a possible transfer, but a *MultiPopA* makes a transfer unnecessary even though *PushA* has paid for it.