Exercise 1:

会计方法:

每次执行操作的平摊代价为3.

其中 1用于执行每个操作至少都有的为1的基本代价

1作为存款用于该操作后跟着最近的为2的整数幂的操作代价

1存入前面用完存款的操作上

可知 $\forall k \le n$ $\sum_{i=1}^{k} a_k - \sum_{i=1}^{k} c_i \ge 0$.

∴执行大小为n的操作序列的平摊代价总和为3n.

Exercise 2:

聚集分析:

第i次操作的代价为 c_i.

如果 $i = 2^m$, $c_i = i$; 否则 $c_i = 1$.

n次 Flipping-Push() 操作的总代价

$\sum_{i=1}^{n} c_i \le n + \sum_{i=0}^{\lfloor \log n \rfloor} 2^i < n + 2n = 3n$

∴每一操作的平摊代价为 $3n/n = 3$.

会计方法:

每次执行操作的平摊代价为3.

其中: 1用于支付每个压栈的基本操作

1作为存款用于支付堆栈反转的代价

1存入栈中第1个没有存款的数据上

∴执行n次 Flipping-Push() 操作的平摊代价总和为3n.

势能方法:

设堆栈中已有元素数目为 num[S], 下一次执行堆栈反转操作时堆栈大小为 size[S]

则定义 $\Phi(S) = 2 \cdot num[S] - size[S]$

由于 $num[S] \geq size[S]/2$. 故 $\Phi(S) \geq 0$.

故总的平摊代价是实际代价的一个上界.

第 i 次操作

未发生翻转: $c_i = 1 \quad \alpha_i = c_i + \Phi(S_i) - \Phi(S_{i-1}) = 3$

发生翻转: $\alpha_i = c_i + \Phi(S_i) - \Phi(S_{i-1}) = 3$

Exercise 3:

(a) MultiPopA(k): $O(n)$

MultiPopB(k): $O(n)$

Transfer(k): $O(n)$

(b) $\Phi(n, m)$ 定义为栈 A, B 中对象所个数相关的函数

$\Phi(n, m) = 2n + m$. 且 $\Phi(n, m)$ 始终 ≥ 0

对操作的平摊代价:

PushA(x): $\alpha_i = c_i + \Phi(n+1, m) - \Phi(n, m) = 3$

PushB(x): $\alpha_i = c_i + \Phi(n, m+1) - \Phi(n, m) = 2$

MultiPopA(k): $\alpha_i = -\min[k, n] < 0$

MultiPopB(k): $\alpha_i = -\min[k, m] < 0$

Transfer(k): $\alpha_i = c_i + \Phi(n-k', m) - \Phi(n, m+k') = 0$

故总的平摊代价为 $O(n)$

每个操作的平摊代价为 $O(n)/n = O(1)$.