Problem Set 6

Due date: Electronic submission of the pdf file of this homework is due on **3/10/2023 before 11:59pm** on canvas.

Name: Huy Quang Lai

Resources. Cormen, Thomas H., Leiserson, Charles E., Rivest, Ronald L., Stein, Clifford. *Introduction to Algorithms.* The MIT Press.

On my honor, as an Aggie, I have neither given nor received any unauthorized aid on any portion of the academic work included in this assignment. This work is my own. Furthermore, I have disclosed all resources (people, books, web sites, etc.) that have been used to prepare this homework. Looking up solutions is prohibited.

Signature: ____________________________

Make sure that you describe all solutions in your own words, and that the work presented is your own! Typesetting in LaTeX is required. Read the chapter on amortized analysis in our textbook.

Problem 1 (20 points)**.** In the lecture, we discussed a stack with PUSH, POP, and MULTIPOP operations. If the set of stack operations included a MULTIPUSH operation, which pushes k items onto the stack, would the $O(1)$ bound on the amortized cost of stack operations continue to hold?

Solution. The $O(1)$ bound on the amortized cost of stack operations would **NOT** continue to hold.
The time complexity of such a series of operations depends on the number of pushes could be made. Since one `MULTIPUSH` needs $O(k)$ time, performing n `MULTIPUSH` operations would take $O(kn)$ time, leading to amortized cost of $O(k)$.

Problem 2 (20 points)**.** In the lecture, we discussed a k-bit counter with an INCREMENT operation. Show that if a DECREMENT operation were included in the k-bit counter example, n operations could cost as much as $\Theta(nk)$ time.

Solution. In the worst case `INCREMENT` needs to flip all ones to zeros. Example $0111111 \rightarrow 1000000$.
Likewise for `DECREMENT` needs to flip all zeros to ones. Example $1000000 \rightarrow 0111111$. Both of these are both $\Theta(k)$ for k-bit counters.
Therefore, alternating between `INCREMENT` and `DECREMENT` is $\Theta(nk)$

Problem 3 (20 points)**.** Suppose we perform a sequence of n operations on a data structure in which the i-th operation costs i if i is an exact power of 2, and 1 otherwise. Use aggregate analysis to determine the amortized cost per operation.

Solution. In a sequence of n operations there are $\lfloor \log_2 n \rfloor + 1$ exact powers of 2. They are: $2^0, 2^1, 2^2, \cdots, 2^{\lfloor \log_2 n \rfloor}$
The total cost of the cost is a geometric sum:

$$\sum_{i=0}^{\lfloor \log_2 n \rfloor} 2^i = 2^{\lfloor \log_2 n \rfloor + 1} - 1 \leq 2^{\log_2 n + 1} = 2n$$

The rest of the operations are $O(1)$. Of there operations there are at most n.
The total cost of all operations is $T(n) \leq 2n + n = 3n \in O(n)$.
This leads to an amortized cost per operation of $O(1)$.

Problem 4 (20 points)**.** Redo Problem 3 using an accounting method of analysis.

Solution. The actual cost of the ith operations is

$$c_i = \begin{cases} i & \text{if } i = 2^k, k \in \mathbb{Z}^+ \\ 1 & \text{otherwise} \end{cases}$$

Assigning amortized costs as follows:

$$\hat{c_i} = \begin{cases} 2 & \text{if } i = 2^k, k \in \mathbb{Z}^+ \\ 3 & \text{otherwise} \end{cases}$$

Under this definition, when i is an exact power of 2, the operation uses all previously accumulated credit plus one unit of its own amortized cost to pay its true cost. It then assigns the remaining unit as credit.

When i is not an exact power of 2, then the operation uses one unit to pay its actual cost and assigns the remaining two units as credit. This covers the actual cost of the operation.

Therefore, the amortized cost of each operation is $O(1)$.

Problem 5 (20 points)**.** Redo Problem 3 using a potential method of analysis.

Solution. The potential function is defined as follows:

$$\Phi(D_0) = 0 \text{ and } \Phi(D_i) = 2i - 2^{\lfloor \log_2 i \rfloor + 1} + 1 \text{ for } i > 0$$

Since this potential function is always nonnegative, $\Phi(D_i) \geq \Phi(D_0) \forall i$
When i is an exact power of 2, the potential difference is

$$\Phi(D_i) - \Phi(D_{i-1}) = (2i - 2i + 1) - (2(i-1) - (i-1)) = 2 - i$$

Therefore, the amortized cost of the ith operation is

$$\hat{c_i} = c_i + \Phi(D_i) - \Phi(D_{i-1}) = i + 2 - i = 2$$

When i is not an exact power of 2, the potential difference is

$$\Phi(D_i) - \Phi(D_{i-1}) = (2i - i + 1) - (2(i-1) - (i-1)) = 2$$

Therefore, the amortized cost of the ith operation is

$$\hat{c_i} = c_i + \Phi(D_i) - \Phi(D_{i-1}) = 1 + 2 = 3$$

With this result, the amortized cost of each operation is $O(1)$.