CS3230: Assignment for Week 7 Solutions

Due: Sunday, 20th Mar 2022, 11:59 pm SGT.

---

1. Operation $i \in \{1, 2, 4, \ldots, 2^{\lfloor \lg n \rfloor}\}$ cost i, while all other $n - \lfloor \lg n \rfloor - 1$ operations cost 1 each. Hence, the total cost of the n operations is

$$\begin{aligned}
&(1 + 2 + 4 + \cdots + 2^{\lfloor \lg n \rfloor}) + (n - \lfloor \lg n \rfloor - 1) \\
&= (2^{\lfloor \lg n \rfloor + 1} - 1) + (n - \lfloor \lg n \rfloor - 1) \\
&\leq 2n + n = 3n.
\end{aligned}$$

   Dividing by n, we find that the amortized cost is $\Theta(1)$.

2. As discussed in lecture, the 0-th bit gets flipped n times, the 1st bit gets flipped $\lfloor n/2 \rfloor$ times, the 2nd bit gets flipped $\lfloor n/4 \rfloor$ times, and so on. Hence, the total cost of n increment operations is

$$f(n) = n \cdot 1 + \lfloor n/2 \rfloor \cdot 2 + \lfloor n/4 \rfloor \cdot 4 + \cdots + 1 \cdot 2^{\lfloor \lg n \rfloor}.$$

   There are $\lfloor \lg n \rfloor + 1$ terms in this sum, each of which is at most n. Moreover, if we let m be the largest power of 2 that is at most n, then $f(n)) \geq f(m) = m(\lg m + 1) \geq \frac{n}{2} \lg n$. Hence, $f(n) = \Theta(n \lg n)$, and so the amortized cost is $\Theta(\lg n)$.

3. As in the hint, we keep a pointer to the highest-order 1-bit, move this pointer one position to the left if the highest-order 1-bit changes after an INCREMENT operation, and move the pointer to a dummy position to the right of the lowest-order bit. When we do a RESET, we reset the pointer bit and all bits to its right to 0, before moving the pointer.

   We use the accounting method. Charge an amortized cost of US$2+SG$2 for each INCREMENT operation,[1] and SG$1 for each RESET operation. Since these costs are constant, it suffices to show that the total amortized cost is always an upper bound of the total actual cost.

[1] Here I use two currencies to differentiate between the money saved for a later INCREMENT operation and the money saved for a later RESET operation.

When we do an INCREMENT, we pay US$1 to change a bit from 0 to 1, and leave US$1 with this bit to use when it is changed back to 0 in a later INCREMENT. Moreover, we pay SG$1 to check (and, if necessary, update) the pointer, and leave SG$1 with this bit when it is later examined (and possibly changed to 0) by RESET. Hence, we have enough money to pay for both types of operations.