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: Ayushri Jain

Resources.

Cormen, Leiserson, Rivest, Stein: Introduction to Algorithms, 3rd edition, The MIT Press, 2009 (or 4th edition)

https://www.geeksforgeeks.org/introduction-to-amortized-analysis/

https://www.geeksforgeeks.org/potential-method-in-amortized-analysis/

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: AJain

Problem 1 (20 points).

Solution. The stack operations that we discussed in the lecture were PUSH, POP and MULTIPOP to which we assigned amortized cost of 2, 0 and 0 respectively, meaning PUSH pays for future POP and MULTIPOP operations. All these operations have linear time complexity. If we include a MULTIPUSH operation and since 1 element push takes 1 unit time, k elements push will take k time. So, performing n MULTIPUSH operations will take nk time. This MULTIPUSH should pay for future POP and MULTIPOP operations, therefore per element it should be charged 2. Thus, amortized cost of MULTIPUSH operations will become $O(2kn)/n = O(k)$, i.e. it is proportional to k = number of items pushed which is not $O(1)$.

Problem 2 (20 points).

Solution. We need to consider a k-bit counter which has both INCREMENT and DECREMENT operations. In the worst-case scenario, we will have to perform INCREMENT to set all bits to 1 and then perform DECREMENT operation to set all bits to 0. To perform this operation, we need to flip all k bits from 1 to 0. Since we don't know which bit is set to 1, we need to perform k DECREMENT operations, one for each bit. Each DECREMENT operation will take $\Theta(k)$ time, because we need to propagate the borrow from least significant bit to most significant bit. Same will be the case of INCREMENT operations, only difference that we travel in reverse order but the time will be $\Theta(k)$. Therefore, the total time to perform any n number of combination of INCREMENT and DECREMENT operations is $\Theta(nk)$ in the k-bit counter.

Problem 3 (20 points).

Solution. Let n denote the number of operations, so upto n, there will $\lfloor \log_2 n \rfloor + 1$ exact powers of 2 and if i denotes such a power of 2, i^{th} operation will cost i. The remaining operations will cost 1 each and number of remaining operations is $n - (\lfloor \log_2 n \rfloor + 1)$. So, total cost of n operations is sum of cost of these remaining operations plus cost of $\lfloor \log_2 n \rfloor + 1$ powers of 2 which is a geometric series. Sum of this geometric series is $= 1 + 2 + 2^2 + \cdots + 2^{\lfloor \log_2 n \rfloor} = 2^{\lfloor \log_2 n \rfloor + 1} - 1$. Sum of remaining operations is $n - (\lfloor \log_2 n \rfloor + 1) * 1 = n - \lfloor \log_2 n \rfloor - 1$. Total cost of n operations $= 2^{\lfloor \log_2 n \rfloor + 1} - 1 + n - \lfloor \log_2 n \rfloor - 1$. This can be simplified as : $2^{\lfloor \log_2 n \rfloor + 1} - 1 + n - \lfloor \log_2 n \rfloor - 1$
$= 2 * 2^{\lfloor \log_2 n \rfloor} + n - \lfloor \log_2 n \rfloor - 2$
$= 2 * n^{\lfloor \log_2 2 \rfloor} + n - \lfloor \log_2 n \rfloor - 2$ ... using logarithm property
$= 2n + n - \lfloor \log_2 n \rfloor - 2$
$= 3n - \lfloor \log_2 n \rfloor - 2$ which is less than $3n$, i.e. cost of n operations is less than $3n$. So, cost per operation is less than 3. Hence, we can say that amortized cost per operation is 3.

Problem 4 (20 points).

Solution. We can assign a cost of 3 to each operation, where the first operation costs 1, giving us a credit of 2. Second operation costs 2, giving us a credit of

2. Third operation costs 1, $\cdots$ and so on, powers of 2 cost i. After the 2^i-th operation, each of the $2^i - 1$ operations following it costs 1. As we pay 3 for each operation, we use 1 credit and gain 2 credits for each of the next $2^i - 1$ operations, giving us a total credit of $2 * (2^i - 1) = 2^{i+1} - 2$. When we perform the (2^{i+1})-th operation, we first pay 3, thus we have $2^{i+1} - 2 + 3 = 2^{i+1} + 1$ accumulated credits. These credits can be used for the actual cost of 2^{i+1}, leaving us with 1 credit. Therefore, we always have a non-negative credit after each operation. Hence, we can say that our assignment of amortized cost of 3 per operation is correct.

Problem 5 (20 points)**.**

Solution. Based on our observation in the last 2 questions, we can think about a potential function which denotes the credit accumulated after i^{th} operation. Let $\Phi(D_i)$ denote our potential function, then

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

Since we observed before that the credit accumulated is always non-negative, so $\Phi(D_i) \geq 0 = \Phi(D_0)$ for all $i > 0$. When i is exact power of 2, the potential difference is $\Phi(D_i) - \Phi(D_{i-1}) = (2i - 2i + 1) - (2(i-1) - i + 1) = 2 - i$ (because $2^{\lfloor \log_2 i \rfloor + 1} = 2 * 2^{\log_2 i} = 2i$). This is also true for $i = 1$. Actual cost c_i for such operation is i. Thus, the amortized cost of i^{th} operation is $\hat{c_i} = c_i + \Phi(D_i) - \Phi(D_{i-1}) = i + 2 - i = 2$.
But when i is not an exact power of 2, the potential difference is $\Phi(D_i) - \Phi(D_{i-1}) = (2i - i + 1) - (2(i-1) - i + 1) = 2$ because of the floor function. Actual cost in such operation is 1. Thus, the amortized cost of this type of i^{th} operation is $\hat{c_i} = c_i + \Phi(D_i) - \Phi(D_{i-1}) = 1 + 2 = 3$.
Based on above 2 observations, we can say that the amortized cost per operation is 3.