Assignment 7

Jixuan Ruan PB20000188

November 18, 2022

1. (a) Let's consider the condition when $n = 2^k$.

$$\sum_{i=1}^{n} c_i = \sum_{j=1}^{k} 2^{j-1} + 2^j = 3 \cdot (2^k - 1) = 3(n-1)$$

So the amortized time complexity is $O(1)$.

(b) Let's consider the condition when $n = 2^k$.
We can assumed the amortized cost of the 2^kth operation is $3 \times 2^{k-1}$ and the amortized time complexity of the other operation is 0.

$$\sum_{i=1}^{n} \hat{c}_i = \sum_{j=1}^{k} 3 \times 2^{j-1} = 3(n-1)$$

So the amortized time complexity is $O(1)$.

(c) We assume $\Phi(D_i) = 2^{1+\lfloor \lg i \rfloor} - i$ when $i \geq 1$, and $\Phi(D_0) = 0$.

$$\hat{c}_i = 2(i = 1)$$
$$\hat{c}_i = 2^k + 2^k - 1 = 2 \times n - 1(i = 2^k, k \geq 1)$$
$$\hat{c}_i = 0(others)$$
$$\sum_{i=1}^{n} \hat{c}_i = 2^{\lfloor \lg n \rfloor + 2} - 2 - \lg n$$

So the amortized time is still $O(1)$.

2. We combine two stacks to a queue and we maintain another two stacks to record the minimum element.
The ENQUEUE contains two operations.
1.Push the new element to s_1.
2.$ele = smin1.top()$.if the new element is smaller than ele, then we push the new element into the smin1, otherwise we push the previous ele into smin1.

The DEQUEUE operation. There are two conditions.
1. If s_2 is empty, then we dump the elements in s_1 into s_2. Later we push the elements in smin1 into smin2 as the step in ENQUEUE's second operation.
2. If s_2 is not empty, then we simply pop the top element in both s_2 and $smin2$.

The FINDMIN contain only one operation.
1.Print the smaller element between $smin1$'s top element and $smin2$'s top element.

We assume the $\Phi(D_i) = 3 \times s_1.size$.
Let's consider the c_i in ENQUEUE operation.

$$\hat{c_i} = c_i + 3 = 6$$

Let's consider the first condition in DEQUEUE operation.

$$\hat{c_i} = c_i - 3 \times s1.size = 0$$

Let's consider the second condition in DEQUEUE operation.

$$\hat{c_i} = c_i + 0 = 2$$

So $\sum_{i=1}^{n} c_i < 6n$. We can see the amortized time cost is $O(1)$.

3. At each step we will choose the subset S_i whose inclusion covers the maximum number of uncovered elements. We can see the time complexity is $O(kmn)$.Since in every iteration we find the S_i in the left sets and delete the new covered elements in the left sets, the average cost of this operation is $O(mn+m)$. There is k iterations, so the time complexity is $O(kmn)$ which is polynomial.

   Proof for the approximation:
   We assume the ith new added elements' quantity is a_i, and the size of the set in the ith iteration is s_i. According to the design of the greedy algorithm $a_i \geq \frac{1}{k}(OPTIMUM - s_{k-1})$.

$$\begin{aligned}
\frac{1}{k}(OPTIMUM - s_{k-1}) &\leq s_k - s_{k-1} \\
\frac{1}{k}(OPTIMUM - s_{k-1}) &\leq (OPTIMUM - s_{k-1}) - (OPTIMUM - s_k) \\
(1 - \frac{1}{k})(OPTIMUM - s_{k-1}) &\geq (OPTIMUM - s_k)
\end{aligned}$$

   We already know that $s_0 = 0$.

$$\begin{aligned}
(1 - \frac{1}{k})^k OPTIMUM &\geq (OPTIMUM - s_k) \\
s_k &\geq (1 - (1 - \frac{1}{k})^k) OPTIMUM
\end{aligned}$$

4. I cannot solve it.