CSC263, Problem Set 3 Q2

Youngsin Ryoo, [redacted]

Question 2

a) Let's say the length of the sequence is n. Since TASK1 is consecutive and each TASK1 operation O_i after the first is twice as long as the previous TASK1 operations, we have: $T_1(n) = 1 + 2 + 4 + 8 + \cdots + 2\cdot$ (run time of $(O_{n-1})) = 2^n - 1$. However, we know there are constant numbers of TASK2 in between. So, $T_1(n) \leq 2^{\lceil \frac{n}{2} \rceil} - 1$. TASK2 takes constant time and there are constant times of TASK2, so $T_2(n) = (n - \frac{n}{2})k$, where k represents constant running time for TASK2. Then, we have:

$$T_1(n) + T_2(n) = 2^{\lceil \frac{n}{2} \rceil} - 1 + (n - \frac{n}{2})k$$

Now, we have to divide this by n, which is the number of operations. Then, we get:

$$\begin{aligned}\frac{2^{\lceil \frac{n}{2} \rceil} - 1 + (n - \frac{n}{2})k}{n} &= \mathcal{O}(\frac{2^{\lceil \frac{n}{2} \rceil}}{n}) \\ &\quad (k \text{ is asymptotically bounded as it is constant time}) \\ &= \mathcal{O}(2^{\frac{n}{2}})\end{aligned}$$

Therefore, we have a running time complexity of $\mathcal{O}(2^n)$.

b) Let's say the length of the sequence is n. Consider when $n = 4, 9, 19, \cdots$. When $n = 4$, we get 2 TASK1's and 2 TASK2'S. When $n = 9$, we get 3 TASK1's and 6 TASK2's. When $n = 16$, we get 4 TASK1's and 12 TASK2's. From this pattern, we can conclude that $T_1(n) = 2^{\lfloor \sqrt{n} \rfloor} - 1$, and $T_2(n) = (n - \sqrt{n}) \cdot k$, where k represents constant running time for TASK2. Putting $T_1(n)$ and $T_2(n)$ together, we get:

$$T_1(n) + T_2(n) = 2^{\lfloor \sqrt{n} \rfloor} - 1 + (n - \sqrt{n}) \cdot k$$

Now, we have to divide this by n, which is the number of operations. Then, we get:

$$\begin{aligned}\frac{2^{\lfloor \sqrt{n} \rfloor} - 1 + n - \sqrt{n}}{n} &= \mathcal{O}(2^{\sqrt{n}}) \quad (k \text{ is asymptotically bounded as it is constant time}) \\ &= \mathcal{O}(2^n).\end{aligned}$$

Therefore, we have a running time complexity of $\mathcal{O}(2^n)$.

c) Let's say the length of the sequence is n. The worst case scenario for this case is when there is initially one TASK2 (a.k.a O_2), followed by TASK1 right after, because this sequence will have the most number of TASK1's. So, in other words, the worst case would look like $T_1 + T_2 + T_1 + T_2 + T_2 + T_1 + \cdots + T_1$. In this case, there would be at most $\lfloor log(n) \rfloor$ number of TASK1 because each time TASK2 is in between TASK1's, the number of TASK2 doubles. Since the running time of each TASK1 is twice as long as the previous TASK1 operation, we will have $T_1(n) = 2^{\lfloor log(n) \rfloor} - 1$ running time for TASK1. On the other hand, the running time of TASK2 will be $T_2(n) = (n - \sqrt{n}) \cdot k$, where k represents the constant running time of TASK2. Putting $T_1(n)$ and $T_2(n)$ together, we get:

$$\begin{aligned} T_1(n) + T_2(n) &= 2^{\lfloor log(n) \rfloor} - 1 + (n - log(n)) \cdot k \\ &= n - 1 + (n - log(n)) \cdot k \end{aligned}$$

Now, we have to divide both by n, which is the number of operations. Then, we get:

$$\begin{aligned} &= \frac{n - 1 + (n - log(n)) \cdot k}{n} \\ &= \mathcal{O}(\frac{n}{n}) && (k \text{ is asymptotically bounded as it is constant time}) \\ &= \mathcal{O}(1). \end{aligned}$$

Although the worst case running time is $\mathcal{O}(n)$, the amortized running time complexity for each operation is $\mathcal{O}(1)$.