Solutions to Week 12 Exercises

Ibrahim Chehab

April 9, 2024

1 Questions from FSG Slides

1.1 The Tightest Bound

Peace and d.aki are arguing over the tightest upper-bound for the following function. Peace argues that it's $\mathcal{O}(n^2)$ while d.aki argues that it's $\mathcal{O}(n^3)$. Given the following function, who is correct? Justify your answer.

1.1.1

```
def mystery(n: int):
L = []
for i in range(n):
    for j in range(min(50, n)):
        L.insert(0, j)
```

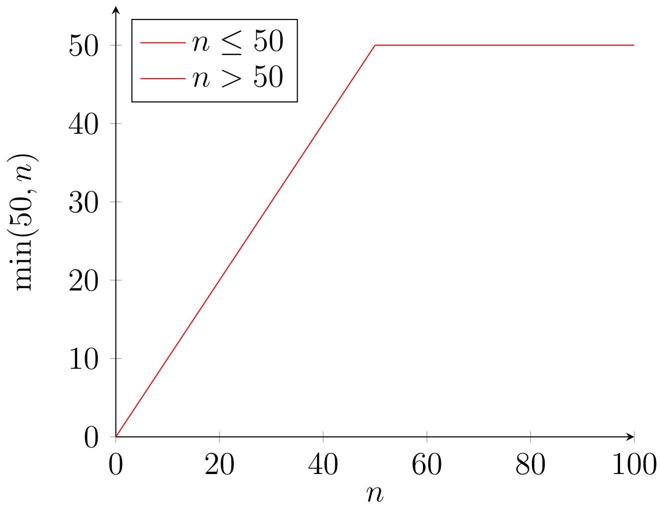
Solution: To solve this question, we will first begin by analyzing the time complexity of the inner loop. The inner loop runs for $\min(50, n)$ iterations. This means that the inner loop runs for 50 iterations if $n \geq 50$ and n iterations if $n < 50$.

Inside this loop, we have an $\mathcal{O}(n)$ operation, which is the `insert` operation. Then, encapsulating all of this, we have an outer loop that runs for n iterations.

With this information, we will construct a piecewise function $T(n)$ that represents the time complexity of the function `mystery`.

$$T(n) = \begin{cases} 50n^2 & \text{if } n \geq 50 \\ n^3 & \text{if } n < 50 \end{cases}$$

Notice, for sufficiently large n (i.e. $n \geq 50$), the complexity of the second loop drops out. This can be seen when graphing the function `min(50, n)`.



Notice, for sufficiently large n, the inner-most loop runs for 50 iterations, making it run in constant time (i.e: regardless of the value of n, the inner loop runs for 50 iterations).

This means that for sufficiently large n, the time complexity of the function `mystery` is $\mathcal{O}(50n^2) \implies \mathcal{O}(n^2)$. Therefore, Peace is correct.

With that said, however, d.aki is still correct to some degree. We know that for our function T to be an element of $\mathcal{O}(n^3)$, there must exist some $c > 0$ and $n_0 > 0$ such that $T(n) \leq cn^3$ for all $n \geq n_0$. Since our function $T \in \mathcal{O}(n^2)$, it is vacuously true that $T(n) \leq cn^3$ for all $n \geq n_0$ for some $c > 0$ and $n_0 > 0$. Therefore, d.aki is correct in that the function is also $\mathcal{O}(n^3)$. However, this is **not** the tightest upper-bound for the function.

Therefore, while both are correct, Peace is more correct in this case.

1.1.2

```python
def mystery2(n: int):
    L = []
    for i in range(n):
        for j in range(max(50, n)):
            L.insert(0, j)
```

Solution: As the question suggests, we will modify our piecewise $T(n)$ function to account for the change in the inner loop. The inner loop now runs for $\max(50, n)$ iterations. This means that the inner loop runs for 50 iterations if $n < 50$ and n iterations if $n \geq 50$.

This translates to the following

$$T(n) = \begin{cases} 50n^2 & \text{if } n < 50 \\ n^3 & \text{if } n \geq 50 \end{cases} \tag{1}$$

Notice that now for sufficiently large n, the time complexity of the function `mystery2` is $\mathcal{O}(n^3)$. Therefore, d.aki is correct in this case.

Further, unlike the previous function, the function `mystery2` is not $\mathcal{O}(n^2)$. This is because $\mathcal{O}(n^3) \not\subseteq \mathcal{O}(n^2)$. Therefore, Peace is incorrect in this case.

1.2 FindMiiComplexity

What is the worst-case asymptotic complexity of the following function? Justify your answer.

```python
def foo(x: int):
    n = x
    for i in range(50):
    if n <= 50:
        for j in range(n):
        x += 1
    if n <= 100:
        for j in range(n):
        if n <= 75:
            for k in range(n):
            x += 1
```

Solution:

We will follow a similar approach to the previous questions. We will analyze the time complexity of the inner loops and construct a piecewise function $T(n)$ that represents the time complexity of the function `foo`.

The outermost loop runs exactly 50 times, as it is constant and not tied to any function call or input. Creating T_1, T_2, T_3 (Where T_n is the nth loop in the function) to represent the time complexity of the inner loops, we have the following:

$$T_1(n) = 50$$

$$T_2(n) = \begin{cases} n & \text{if } n \leq 50 \\ 0 & \text{if } n > 50 \end{cases}$$

$$T_3(n) = \begin{cases} n & \text{if } n \leq 100 \\ 0 & \text{if } n > 100 \end{cases}$$

$$T_4(n) = \begin{cases} n & \text{if } n \leq 75 \\ 0 & \text{if } n > 75 \end{cases}$$

We can see that for sufficiently large n, none of the outer-loops run, leaving only the outermost loop to run 50 times. This means that the time complexity of the function `foo` is $\mathcal{O}(50) \implies \mathcal{O}(1)$. Therefore, the worst-case asymptotic complexity of the function `foo` is $\mathcal{O}(1)$.

Homework Questions

I can't post solutions for the Exam questions I assigned since those are copyrighted and I don't feel like getting sued.

I won't post the solutions for the "unofficial" homework since it's outside of the scope of the course, and you won't see anything like it until CSC236.