CSC110 Lecture 22: Properties of Asympyotic Growth and Basic Algorithm Running Time

Hisbaan Noorani

November 03, 2020

Contents

1 Exercise 1: Properties of asymptotic growth

1. Recall the following definition:

   Let $f, g : \mathbb{N} \to \mathbb{R}^{\geq 0}$. We can define the **sum of f and g** as the function $f + g : \mathbb{N} \to \mathbb{R}^{\geq 0}$ such that

   $$(f+g)(n) = f(n) + g(n), \qquad \text{for } n \in \mathbb{N}$$

   For example, if $f(n) = 2n$ and $g(n) = n^2 + 3$, then $(f+g)(n) = 2n + n^2 + 3$.

   Consider the following statement:

   $$\forall f, g : \mathbb{N} \to \mathbb{R}^{\geq 0},\ g \in \mathcal{O}(f) \implies f + g \in \mathcal{O}(f)$$

   In other words, if g is Big-O of f, then $f + g$ is no bigger than just f itself, asymptotically speaking.

   (a) Rewrite this statement by expanding the definition of Big-O (twice!). Use subscripts to help keep track of the variables. This is a good exercise in writing a complex statement in predicate logic, and will help with writing the proof in the next part.
   $\forall f, g : \mathbb{N} \to \mathbb{R}^{\geq 0}, [\exists c_1, n_1 \in \mathbb{R} \text{ s.t. } \forall n \in \mathbb{N}, n \geq n_1 \implies g(n) \leq c_1 \cdot f(n)]$
   $\implies [\exists c_2, n_2 \in \mathbb{R} \text{ s.t. } \forall n \in \mathbb{N}, n \geq n_2 \implies (f+g)(n) \leq c_2 \cdot f(n)]$

   (b) Prove this statement.
   Hint: This is an implication, so you're going to *assume* that $g \in \mathcal{O}(f)$, and you want to *prove* that $f + g \in \mathcal{O}(f)$.
   (Pf:)
   Let $f, g : \mathbb{N} \to \mathbb{R}^{\geq 0}$
   Assume $g \in \mathcal{O}(f)$. This means $\exists c_1, n_1 \in \mathbb{R}$ s.t. $\forall n \in \mathbb{N}, n \geq n_1 \implies g(n) \leq c_1 \cdot f(n)$
   Prove $f + g \in \mathcal{O}(f)$. This means $\exists c_2, n_2 \in \mathbb{R}$ s.t. $\forall n \in \mathbb{N}, n \geq n_2 \implies (f+g)(n) \leq c_2 \cdot f(n)$
   Take $n_2 = n_1$
   Take $c_2 = c_1 + 1$
   Let $n \in \mathbb{N}$
   Assume $n \geq n_2$

Prove $(f+g)(n) \leq c_2 \cdot f(n)$
Since $n_2 = n_1$, we know $n \geq n_1$ thus $g(n) \leq c_1 \cdot f(n)$.

$$\begin{aligned}
(f+g)(n) &\leq c_2 \cdot f(n) \\
f(n)+g(n) &\leq (c_1+1)\cdot f(n) && \text{(since } c_2 = c_1+1\text{)} \\
f(n)+g(n) &\leq c_1 \cdot f(n) + f(n) \\
g(n) &\leq c_1 \cdot f(n) \quad \blacksquare && \text{(subtract } f(n) \text{ from both sides)}
\end{aligned}$$

We have arrived at our assumption, $g(n) \leq c_1 \cdot f(n)$, and thus we have proven $f+g \in \mathcal{O}(f)$ as needed.

2 Exercise 2: Analysing running time (for loops)

Analyse the running time of each of the following functions, in terms of their input length n. Keep in mind these three principles for doing each analysis:

- For each for loop, determine the *number of iterations* and the *number of steps per iteration*.
- When you see statements in sequence (one after the other), determine the number of steps for each statement separately, and then add them all up.
- When dealing with nested loops, start by analyzing the inner loop first (the total steps of the inner loop will influence the steps per iteration of the outer loop).

1. Number of iterations: n

   Number of steps per iteration: 2

   $RT_{f_1} = 2 \cdot n$

   $RT_{f_1} \in \Theta(n)$

```python
    def f1(numbers: List[int]) -> None:
        for number in numbers:
            print(number * 2)
```

1. 1st loop

   Number of iterations: n

   Number of steps per iteration: 2

   2nd loop

   Number of iterations: 10

   Number of steps per iteration: 3

   $RT_{f_2} = 2 \cdot n + 3 \cdot 10$

   $RT_{f_2} \in \Theta(n)$

```python
    def f2(numbers: List[int]) -> int:
        sum_so_far = 0
        for number in numbers:
            sum_so_far = sum_so_far + number

        for i in range(0, 10):
            sum_so_far = sum_so_far + number * 2

        return sum_so_far
```

2. Inner loop

   Number of iterations: n

   Number of steps per iteration: 2

   Outer loop

   Number of Iterations: $n^2 + 5$

   Number of steps per iteration: $2n$

   $RT_{f_3} = 2n^3 + 10n$

   $RT_{f_3} \in \Theta(n^3)$

```python
    def f3(numbers: List[int]) -> int:
        for i in range(0, len(numbers) ** 2 + 5):
            for number in numbers:
                print(number * i)
```

3 Additional exercises

Review the properties of Big-O/Omega/Theta we covered in lecture today, and try proving them! You should be able to prove all of them except (5) and (6) of the Theorem on the growth hierarchy of elementary functions.