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

Print this handout

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:[1]

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

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 kee 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.

$\forall f, g : \mathbb{N} \to \mathbb{R}^{\geq 0},\ \big(\exists c_0, n_0 \in \mathbb{R}^+,\ \forall n \in \mathbb{N},$
$\quad n \geq n_0 \Rightarrow g(n) \leq c_0 f(n)\big)$
$\Rightarrow \big(\exists c_1, n_1 \in \mathbb{R}^+,\ \forall n \in \mathbb{N},$
$\quad n \geq n_1 \Rightarrow (f(n) + g(n)) \leq c_1 f(n)\big)$

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 $f + g \in \mathcal{O}(f)$.

Let $f, g : \mathbb{N} \to \mathbb{R}^{\geq 0}$. Assume $\exists c_0, n_0 \in \mathbb{R}^+, \forall n \in \mathbb{N}, n \geq n_0 \Rightarrow g(n) \leq c_0 f(n)$

We want to prove that

$\exists c_1, n_1 \in \mathbb{R}^+, \forall n \in \mathbb{N}, n \geq n_1 \Rightarrow (f(n) + g(n) \leq c_1 f(n))$

Let $c_1 = \underline{1 + c_0}$, $n_1 = \underline{n_0}$ and let $n \in \mathbb{N}$.

Assume $n \geq n_1$.

Then, since $n \geq n_1$, $n \geq n_0$ and it follows that $g(n) \leq c_0 f(n)$. Add $f(n)$ to both sides to get $f(n) + g(n) \leq f(n) + c_0 f(n)$

$= (1 + c_0) f(n)$

Exercise 2: Analysing running time (for loops)

$= c_1 f(n)$, as req'd.

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

rough work:

- # iterations is len(numbers)

· # steps per iteration is 1.

Let n be the length of the input list numbers.

Each iteration of the loop can be counted as a single step, since nothing in the loop depends on the size of the list numbers. The loop iterates n times. Hence, the total number of basic operations is $n \cdot 1$ and so $RT_{f1}(n) = n \in \Theta(n)$.

2.

```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 + i * 2

    return sum_so_far
```

rough work: 4 blocks to consider

- assignment stmt 1 step
- for loop:
  - # iterations - len(numbers)
  - steps per iteration: 1
- for loop:
  - # iterations - 10
  - steps per iteration: 1
- return stmt 1 step

3.

```python
def f3(numbers: list[int]) -> None:
    for i in range(0, len(numbers) ** 2 + 5):
        for number in numbers:
            print(number * i)
```

Q3:

rough work:

inner loop:

iterations: len of list numbers

steps per iteration: 1

outer loop:

iterations: $\text{len(numbers)}^2 + 5$

steps per iteration: $\text{len(numbers)} \cdot 1$

Formal running time analysis:

Let n be the length of the list numbers.

The inner loop runs n times, and each step is just a single step. But the inner loop is itself repeated, since it is inside another loop.

The outer loop runs n^2+5 times, and each of its iterations takes $n \cdot 1$ steps.

Thus the total number of basic operations is: $(n^2+5) \cdot n$

$$= n^3 + 5n$$

Hence $RT_{f3}(n) = n^3 + 5n \in \Theta(n^3)$.

L Q2 continued

Formal running time analysis:

Let n be the length of the input list numbers.

The assignment and return step each take 1 step.

The first loop iterate n times with each iteration taking 1 step, for a total of n steps.

The second loop iterate 10 times with each iteration taking 1 step, for a total of 10 steps.

Thus, the ~~total~~ number of basic operations is $2 + n \cdot 1 + 10$, so

$$RT_{f2}(n) = n + 12 \in \Theta(n).$$

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 *Elementary function growth hierarchy theorem*.

---

1. This statement is a simpler form of the more general "Sum of Functions" Theorem we saw in lecture.↩