Algorithms - Day 2

Instructor: Pat Devlin — prd41@math.rutgers.edu

Summer, 2016

Asymptotic notation

Etymology of *algorithm*

Roman numerals were used in Europe for a shockingly long time[1]. For example, the number 1448 was written as MCDXLVIII, which (combining with modern notation) can be broken down as

$$MCDXLVIII = M + (CD) + (XL) + V + III = 1000 + (-100 + 500) + (-10 + 50) + 5 + 3.$$

In addition to be difficult to read, Roman numerals are also very difficult to work with [try adding MCDXLVIII and DLXVIII (or try multiplying them!)]. Fortunately, during the centuries while Europe was intellectually stagnant the rest of the world was doing great math. The positional system we now use today (1448 + 568 = 2016) was invented in India and brought to the west through a textbook written by a Persian mathematician named Al Khwarizmi (c. 780 – c. 850). The book was translated into Latin as *Algoritmi de numero Indorum* (Algoritmi on the numbers of the Indians). Those trained in this text were able to do the complex numerical procedures described such as add, multiply, and divide numbers (and even compute $\sqrt{23}$). These numerical wizards were known as *algorists*, and their craft was called *algorism*.

Big-O notation

Let's recall the search problem and its algorithms from yesterday.

Example 1 (Search problem) Pat has a secret list of n distinct real numbers. Each number is covered up so that you can't see it, but Pat promises that the list is sorted by size (increasing from left to right). You know that the number 100 is *somewhere* in that list, and your goal is to find it. You are allowed to uncover any number you want, but every time you do this, you need to give Pat $1.

```
Silly_Search:
begin
  Uncover all of the numbers.
  Find the number we want.
end
```

```
Better_Search:
begin
  Uncover the numbers one at a time.
  If we find the number we want, then stop. Otherwise, uncover the next number.
end
```

[1]Portions of this lesson (i.e., this etymology) are adapted from the (great and free) online text *Algorithms* by Dasgupta, Papadimitriou, and Vazirani.

```
Binary_Search:
begin
  Uncover the middle number, and call its value x
  if x = 100 then we found it, so STOP
  if x < 100 then start over, but only use the half of the list bigger than x
  if x > 100 then start over, but only use the half of the list smaller than x
end
```

Question 2 How many numbers do these algorithms need to uncover as a function of n? (As *always*, we are only concerned about how many in the *worst-case* scenarios.) Determine the costs of these algorithms for the values $n = 10^3$ and $n = 10^6$.

When we think about comparing two algorithms, we ultimately need to compare two functions to see which is "bigger" than the other. When computer scientists do this, we use big-O notation.

Definition 3 Let $f(n)$ and $g(n)$ be two functions. We say $f(n) = \mathcal{O}(g(n))$ to mean that there is a constant $M > 0$ such that $f(n) \leq Mg(n)$ for all sufficiently large n. (We read this as "$f(n)$ is big-O of $g(n)$")

You want to think of the statement $f(n) = \mathcal{O}(g(n))$ as something like $f(n) \lessapprox g(n)$.

Saying $f(n) = \mathcal{O}(g(n))$ essentially means that $g(n)$ grows at least as fast as $f(n)$. Statements like these are about the *asymptotic growth* of the functions.

Example 4 If $f(n) = 15n^2 + 10^{50}$, then $f(n) = \mathcal{O}\left(n^2\right)$ because the inequality $f(n) \leq 16n^2$ holds for all sufficiently large values of n. (Notice that this inequality actually *doesn't* work if $n < 10^{25}$, but when we use big-O notation, we only care about what's happening when n is [very] large.)

Example 5 Using big-O notation can help us better understand complicated expressions. For instance,

$$\sqrt{n^6 + 2n^4 + 3n^3 + 8\sqrt{n} + 1000} = \mathcal{O}\left(n^3\right).$$

Example 6 Argue each of the following:

- $n^3 + 3n^2 - 1 = \mathcal{O}\left(n^3\right)$
- $n^3 + 3n^2 - 1 = \mathcal{O}\left(n^{10}\right)$
- $\dfrac{n^5 + 6}{n + 20} = \mathcal{O}\left(n^4\right)$
- $10^{-n} = \mathcal{O}\left(1\right)$
- $15(n!) = \mathcal{O}\left(n^n\right)$

Question 7 Sort the following functions in terms of asymptotic growth. (i.e., arrange them in a list $f_1, f_2, f_3, \ldots$ so that $f_1 = \mathcal{O}\left(f_2\right)$ and $f_2 = \mathcal{O}\left(f_3\right)$, et cetera)

$$n, \lg(n), n^2, 2^n, n^{100}, 1, 3^n, n^3, \lg^2(n), \sqrt{n}, n!, n^n, 2^{2n}, n\lg(n)$$

A more precise way to describe asymptotic growth is as follows.

Definition 8 Let $f(n)$ and $g(n)$ be two functions. We say $f(n) = \Theta(g(n))$ to mean that there are constants $M_1 > 0$ and $M_2 > 0$ such that $M_1 g(n) \leq f(n) \leq M_2 g(n)$ for all sufficiently large n. (We read this as "$f(n)$ is theta of $g(n)$")

You want to think of the statement $f(n) = \Theta(g(n))$ as something like $f(n) \approx g(n)$.

Proposition 9 *If* $f(n) = \Theta(g(n))$ *then* $g(n) = \Theta(f(n))$.

Proof:

Proposition 10 $f(n) = \Theta(g(n))$ *if and only if* $f(n) = \mathcal{O}(g(n))$ *and* $g(n) = \mathcal{O}(f(n))$.

[Proof left as an exercise.]

When you're studying algorithms, a statement like $f(n) = \Theta(g(n))$ means that f and g are essentially interchangeable. This is often very convenient.

Example 11 If $f(n) = 3n^2 + 1000n - 10000$, then $f(n) = \Theta(n^2)$.

Question 12 For each of the following, prove $f(n) = \Theta(g(n))$ or give an argument why this is not the case.

- $f(n) = \sqrt{n^6 + 2n^4 + 3n^3 + 8\sqrt{n} + 1000}$ and $g(n) = n^3$
- $f(n) = n^3 + 3n^2 - 1$ and $g(n) = n^3$
- $f(n) = n^3 + 3n^2 - 1$ and $g(n) = n^{10}$
- $f(n) = 10^{-n}$ and $g(n) = 1$
- $f(n) = \log_2(n)$ and $g(n) = \log_{31}(n)$

Question 13 For each of the search algorithms we did yesterday (and at the front of today's notes), find the asymptotic growth for their cost. Use Θ-notation. How does your answer change if Pat changes the price to \$18 per question? What if the price were only \$0.5 per question?

In practice, determining the asymptotic growth of a function usually amounts to using the following simple rules. [Proofs are left as exercises.]

Theorem 14 *Suppose* $f_1(n) = \mathcal{O}(g_1(n))$ *and* $f_2(n) = \mathcal{O}(g_2(n))$. *Then we have*

- $f_1(n) + f_2(n) = \mathcal{O}(g_1(n) + g_2(n))$
- $f_1(n) \times f_2(n) = \mathcal{O}(g_1(n) \times g_2(n))$
- *for all constants* $c > 0$, $(f_1(n))^c = \mathcal{O}(g_1(n)^c)$.

Theorem 15 *Suppose* $f_1(n) = \Theta(g_1(n))$ *and* $f_2(n) = \Theta(g_2(n))$. *Then we have*

- $f_1(n) + f_2(n) = \Theta(g_1(n) + g_2(n))$
- $f_1(n) \times f_2(n) = \Theta(g_1(n) \times g_2(n))$
- $f_1(n)/f_2(n) = \Theta(g_1(n)/g_2(n))$
- *for all constants* $c > 0$, $(f_1(n))^c = \Theta(g_1(n)^c)$.

If we also know that $g_1(n) = \mathcal{O}(g_2(n))$, *then* $g_1(n) + g_2(n) = \Theta(g_2(n))$.

With these rules, determining asymptotic growth becomes extremely easy.

Example 16 Find the asymptotic growth of each of the following.

- $3n^2 + \sqrt{8n^5 + \lg(n) + 18}$
- $9\sqrt[3]{100n! + n^n + n^{37}}$
- $8n^2 \lg(n) + n \lg(n)^2$
- $\dfrac{n^8 + 42}{n^{13} + 4n^4 + 10^7}$

Question 17 Pick an algorithm from yesterday's homework, and imagine how the algorithm would generalize as the size of the problem increases (e.g., n could be the number of bottles, the height of the building, the number of spheres, or the size of the list of numbers). Find the asymptotic growth for the "cost" of the algorithm. (Here "cost" is either the number of tests, drops, or questions required.)