CS:3330 Spring 2017: Handout 1

Review of Common Classes of Running Times

Sriram V. Pemmaraju   Shreyas Pai

We will spend a substantial amount of time in this course analyzing the *running time* of algorithms. The running time of an algorithm is expressed as a mathematical function of the *input size*. This document reviews three of the most common classes of functions that will show up in the running time analysis of algorithms.

Polynomials

A *polynomial* $p(n)$ in n of degree d is a function of the form–

$$p(n) = a_0 + \sum_{i=1}^{d} a_i n^i$$

where the a_i's are the *coefficients* of the polynomial and $a_d \neq 0$. A degree 0 polynomial is just the *constant* function. Degree 1 polynomials are called *linear* functions, degree 2 polynomials are called *quadratic* functions, and degree 3 polynomials are called *cubic* functions.

Example 1: $5n^3 - 10n - 25$ is a cubic polynomial (i.e., a polynomial of degree 3) and the coefficients of this polynomial are $a_0 = -25$, $a_1 = -10$, $a_2 = 0$, and $a_3 = 5$.

We will be interested in the behavior of a polynomial as n becomes very large. In general, the behavior of a function $f(n)$ as n becomes very large is called its *asymptotic* behavior. It is not difficult to see that asymptotically, a polynomial $p(n)$ of degree d behaves just like the term $a_d n^d$ because growth-rate of this term"dominates" the growth-rate of all other terms in the polynomial.

Example 2: Asymptotically, the polynomial $5n^3 - 10n - 25$ behaves like the simpler polynomial $5n^3$. This fact can be precisely stated as:

$$\lim_{n \to \infty} \frac{5n^3 - 10n - 25}{5n^3} = 1.$$

The above-mentioned fact about the asymptotic behavior of polynomials implies that asymptotically, a polynomial of higher degree grows faster than a polynomial of lower degree and will eventually overtake it. More precisely, for any polynomials $p(n)$ of degree d and $q(n)$ of degree d' with $d' > d$, we have

$$\lim_{n \to \infty} \frac{p(n)}{q(n)} = 0. \tag{1}$$

Even though the definition of polynomials only permits degrees d that are non-negative integers, we will also be interested in functions of the form $\sqrt{n} = n^{1/2}$ or $n^{2/3}$ or $n^{3/2}$ in which the exponents are rational numbers. The equation in (1) naturally extends to functions with rational exponents as follows. For any real numbers $d' > d$,

$$\lim_{n \to \infty} \frac{n^d}{n^{d'}} = 0. \tag{2}$$

Example 3: The equation in (2) implies that $\lim_{n\to\infty}(\sqrt{n}/n) = 0$. In other words, the linear function grows asymptotically faster than the function $\sqrt{n}$. Therefore, functions such as $\sqrt{n}$ are referred to as *sublinear* functions. With the explosion of sizes of data sets, we are reaching a point where even algorithms that run in linear time are too slow for some applications and there is a need to look for sublinear time algorithms.

Exponentials

An *exponential function* $f(n)$ is a function of the form a^n, where a is a real number constant. Here, a is referred to as the *base* of the function and n is the *exponent* of the function. Note that if $0 < a < 1$, then a^n is a decreasing function that approaches 0 asymptotically. On the other hand, if $a > 1$, then a^n is an increasing function. For all real numbers $a > 0$, m, and n, exponential functions have the following properties–

$$\begin{aligned}
a^0 &= 1 \\
a^1 &= a \\
a^{-1} &= 1/a \\
(a^m)^n &= a^{mn} \\
(a^m)^n &= (a^n)^m \\
a^m \cdot a^n &= a^{m+n}
\end{aligned}$$

Using these properties we can simplify algebraic expressions and this is useful when we want to compare two different functions.

Example 3: Consider the expressions 4^n and $(\sqrt{2})^{4n}$. We can rewrite both expressions as follows so that they base 2:

$$4^n = (2^2)^n = 2^{2n} \qquad (\sqrt{2})^{4n} = (2^{1/2})^{4n} = 2^{4n/2} = 2^{2n}.$$

This shows that both functions are identical, though this might not have been obvious at first glance.

Using calculus, it is not too difficult to show that *every* exponential function grows faster than *every* polynomial function. A precise statement of this fact is: for all constants a, b where $a > 1$

$$\lim_{n\to\infty} \frac{n^b}{a^n} = 0$$

Example 4: Consider the polynomial function $p(n) = n^{100}$ and the exponential function $f(n) = (1.1)^n$. Even though $p(2) = 2^{100}$ is huge compared to $f(2) = (1.1)^2$, the fact mentioned above tells us that $f(n)$ will eventually overtake $p(n)$.

This implies that any algorithm whose running time is an exponential function a^n, $a > 1$, will eventually take more time (i.e., will be slower) than an algorithm whose running time is a polynomial function. We will discuss this fact more extensively in lecture.

Logarithms

A *logarithm* is just the inverse of the exponential function. That is

$$b^x = a \text{ implies } x = \log_b(a).$$

In other words, $\log_b(a)$ (read as "log of a to the base b")is the quantity that we would raise b to in order to get a. For example, $\log_2(1024) = 10$ because $2^{10} = 1024$ and $\log_{10}(10000) = 4$ because $10^4 = 10000$. The following properties of logarithms are a consequence of its definition and the properties of the exponential function: for all real $a > 0, b > 0, c > 0$, and n and assuming that none of the bases of logarithms are equal to 1:

$$\begin{aligned}
a &= b^{\log_b a} \\
\log_c(ab) &= \log_c a + \log_c b \\
\log_b a^n &= n \log_b a \\
\log_b 1/a &= -\log_b a \\
\log_b a &= \frac{\log_c a}{\log_c b} \text{ (change of base formula)} \\
\log_b a &= \frac{1}{\log_a b} \\
a^{\log_b c} &= c^{\log_b a}
\end{aligned}$$

Again, we can use these properties to simplify expressions containing logarithms.

Example 5: Consider the function $g(n) = 2^{(\log_2 n)^2}$. We can rewrite this as

$$2^{(\log_2 n)^2} = \left(2^{\log_2 n}\right)^{\log_2 n} = n^{\log_2 n}.$$

Note that $n^{\log_2 n}$ grows faster than any polynomial function because the exponent is $\log_2 n$ which is itself a growing function that will exceed any constant. Thus the function $g(n)$ grows faster than any polynomial function.

Example 6: Consider the function $h(n) = \log_3 n$. Using the change of base formula, $h(n) = \log_3 n = \log_2 n / \log_2 3 \approx (0.6309) \log_3 n$. Thus $\log_2 n$ and $\log_3 n$ are functions that are constant multiples of each other.

Example 7: Consider the function $t(n) = n^{1/\log_2 n}$. This function can be simplified as

$$n^{1/\log_2 n} = \left(2^{\log_2 n}\right)^{1/\log_2 n} = 2^{\frac{\log_2 n}{\log_2 n}} = 2.$$

Using calculus, it is not difficult to show that any logarithmic function asymptotically asymptotically grows more slowly than any polynomial function. In fact, any constant power of a logarithmic function grows more slowly than any polynomial function. The precise statement of this is: for any reals d, d', and base $b > 1$:

$$\lim_{n\to\infty} \frac{(\log_b n)^{d'}}{n^d} = 0.$$

Example 8: Thus the function $f_1(n) = (\log_2 n)^{25}$ grows asymptotically more slowly than $f_2(n) = n^2$. This is true despite the fact that $f_1(4) = 2^{25}$ is huge relative to $f_2(4) = 4^2$.

Example 9: Reconsider the function $g(n) = 2^{(\log_2 n)^2}$ from Example 5. We saw there that this function asymptotically grows faster than any polynomial function. Now using the fact that $(\log_2 n)^2$ grows asymptotically more slowly than n, we conclude that $g(n)$ grows asymptotically more slowly than 2^n. Thus the asymptotic growth rate of the function $g(n)$ is "sandwiched" between the class polynomial functions below and the class of exponential functions above.
