Contents of Reflection 6

Part 0. Context: Asymptotic Relations 渐进关系

Asymptotically smaller: $\lim_{n\to\infty}\left(\frac{f(n)}{g(n)}\right) = 0$ tells us that f(n) << g(n)

c is a constant

Asymptotically similar: $\lim_{n\to\infty}\left(\frac{f(n)}{g(n)}\right) = c > 0$ tells us that f(n) ≈ g(n)

Asympototically larger (?) when n → ∞, f(n)/g(n) → ∞ tells us that f(n) >> g(n)

Part 1 The Growth Speed of Various Functions

1<< log n << n << n log n << n^2 << 2^n << n!

Five Extensions: **3^n << 4^n << 5^n…<< k^n, and the following:**

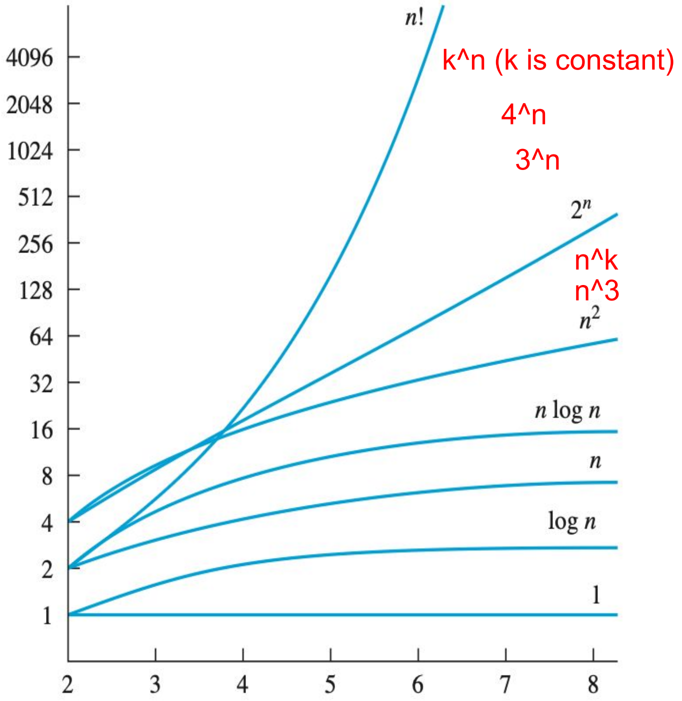
Note:

$2^n << 3^n$ (The ratio $(2/3)^n$ goes to 0 as n goes to infinity

$\log_2 n \approx \log_3 n$ The ratio is a constant.

$\log n << (\log n)^2$

$n \log n << n (\log n)^2$



Part 3. Dominant Term Method

Part 4. Big O

- Definition: We say that f (x) is O(g(x)) if there are constants C and k such that $|f(x)| \leq C|g(x)|$ whenever $x > k$
- C and k: witnesses
- Proof method

Sources:

Rosen 3.2

Leture 12 and its notes

Video 7 - 10

Part 0. Context: Asymptotic Relations 渐进关系

Asymptotically smaller: $\lim_{n\to\infty}\left(\frac{f(n)}{g(n)}\right) = 0$ tells us that f(n) < < g(n)

c is a constant

Asymptotically similar: $\lim_{n\to\infty}\left(\frac{f(n)}{g(n)}\right) = c > 0$ tells us that f(n) ≈ g(n)

Asympototically larger (?) when n → ∞, f(n)/g(n) → ∞ tells us that f(n) >> g(n)

The purpose of introducing this method is **to compare the running time** of different algorithm when they have the same input n (which is a large amount of data). Suppose **n is the input, and f(n) and g(n) are the running time**, then we want to compare f(n) and g(n) goes infinitely large.

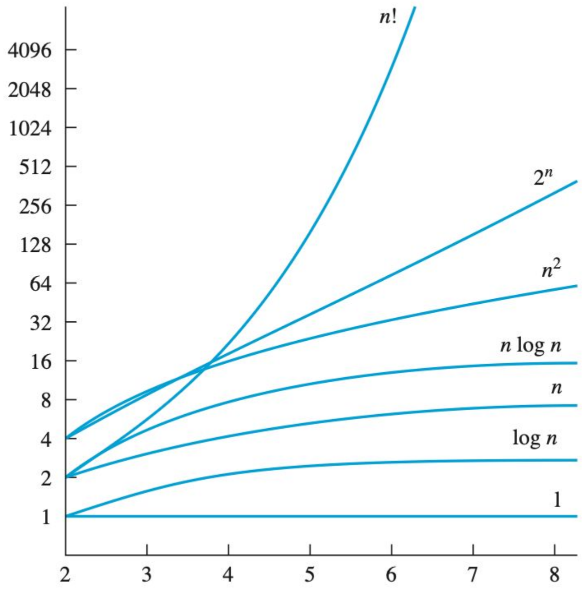
Part 1 The Growth Speed of Various Functions

1. When n becomes infinitely large, we have:

   1<< log n << n << n log n << n^2 << 2^n << n!
   (Rosen 3.2, P 221).

   The above rule can be visualized into the picture on the right side.

- Proof: we have to use calculus to proof this. At this point, we just need to memorize those rules and apply them (Lecture 12).
- To prove n << 2^n when n becomes infinitely large, we have to prove n is O(2^n). (Rosen 3.2, P 211)



2. Extension 1: where to place **k^n**?

- We already have 1<< log n << n << n log n << n^2 << 2^n << n!. Now, we have to consider where to put 3^n, 4^n … k^n; n^3, n^4… n^k.
- The solution is shown in the picture on the right side (Lecture note #12, P4).
- To prove **2^n << 3^n** (video 8):

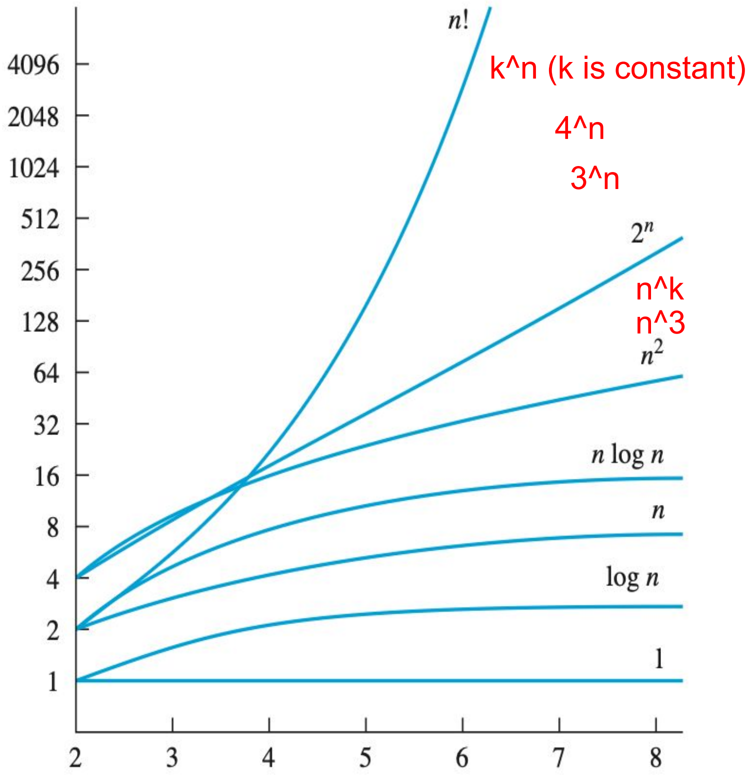
  Let’s say f(x) = 2^n, g(x) = 3^n

  f(x)/g(x) = 2^n/3^n = (⅔)^n

  When n is infinitely large, f(x)/g(x) will be close to 0.

  Therefore, f(x) << g(x), which means 2^n << 3^n.

- Similarly, we can prove **3^n << 4^n << 5^n…<< k^n**



- To prove **$3^n << n!$** (video 8):

  Let's say $f(n) = 3^n$, $g(n) = n!$

  $f(n)/g(n) = 3^n / n!$

  $= 3 * 3 * \ldots * 3 / 1 * 2 * 3 * \underline{4 * 5 \ldots (n-2) * (n-1) * n}$

- After the 3rd number, $4 > 3$, $5 > 3$, $6 > 3 \ldots n > 3$, so every number in the denominator分母 $g(n)$ is larger than that of the molecular分子 $f(n)$.
- The first 3 numbers are too small, so they can be overlooked (according to the dominant term method?).
- So, When $n \rightarrow \infty$, $f(n)/g(n) = 3^n / n! \rightarrow 0$, $f(n) << g(n)$
- Similarly, we can prove **$k^n << n$**.

2. Extension 2: where to place n^k?

n^2 << n^3 << n^4 … << n^k

Proof:

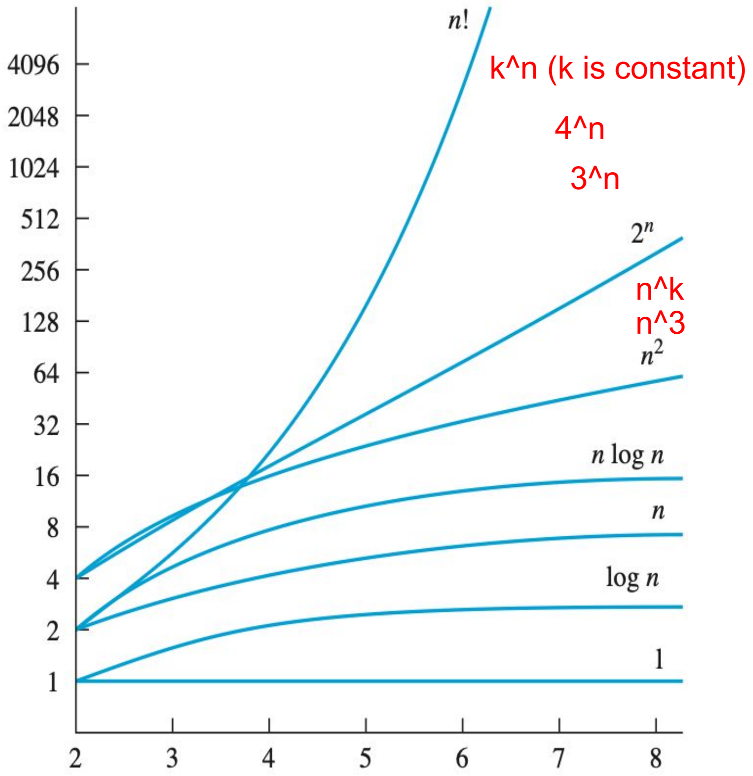
Let f(n) = n^2, g(n) = n^3

f(n)/g(n) = n^2/n^3 = 1/n

When n becomes infinitely large, f(n)/g(n) = 1/n will be close to 0.

Therefore, we have f(n) << g(n), which means n^2 << n^3.

Similarly, we can prove n^2 << n^3 << n^4 … << n^k (k is a constant).



n^k << 2^n

Proof:

EXAMPLE 7 In Section 4.1, we will show that $n < 2^n$ whenever n is a positive integer. Show that this inequality implies that n is $O(2^n)$, and use this inequality to show that $\log n$ is $O(n)$.

Solution: Using the inequality $n < 2^n$, we quickly can conclude that n is $O(2^n)$ by taking $k = C = 1$ as witnesses. Note that because the logarithm function is increasing, taking logarithms (base 2) of both sides of this inequality shows that

$$\log n < n.$$

It follows that

$$\log n \text{ is } O(n).$$

(Again we take $C = k = 1$ as witnesses.)

If we have logarithms to a base b, where b is different from 2, we still have $\log_b n$ is $O(n)$ because

$$\log_b n = \frac{\log n}{\log b} < \frac{n}{\log b}$$

whenever n is a positive integer. We take $C = 1/\log b$ and $k = 1$ as witnesses. (We have used Theorem 3 in Appendix 2 to see that $\log_b n = \log n / \log b$.)

As mentioned before, big-O notation is used to estimate the number of operations needed to solve a problem using a specified procedure or algorithm. The functions used in these estimates often include the following:

$$1,\ \log n,\ n,\ n \log n,\ n^2,\ 2^n,\ n!$$

Using calculus it can be shown that each function in the list is smaller than the succeeding function, in the sense that the ratio of a function and the succeeding function tends to zero as n grows without bound. Figure 3 displays the graphs of these functions, using a scale for the values of the functions that doubles for each successive marking on the graph. That is, the vertical scale in this graph is logarithmic.

3. Extension 3: $\log_2 n \approx \log_3 n$ The ratio is a constant.

Proof:

The change-of-base formula

The formula states that for a logarithm of a number x with base b, you can convert it to a new base c using this relationship:

$$\log_b(x) = \frac{\log_c(x)}{\log_c(b)}$$

.

Applying the formula to $\dfrac{\log_2(n)}{\log_3(n)}$

To solve your problem, you can change both the numerator and the denominator to a common base. A simple choice is to use the natural log ($\ln$, base e) or the common log ($\log$, base 10), since these are available on most calculators.

1. **Change the numerator:** Convert $\log_2(n)$ to the common base 10:

$$\log_2(n) = \frac{\log(n)}{\log(2)}.$$

2. **Change the denominator:** Convert $\log_3(n)$ to the same common base 10:

$$\log_3(n) = \frac{\log(n)}{\log(3)}.$$

3. **Perform the division:** Substitute these expressions back into the original fraction:

$$\frac{\log_2(n)}{\log_3(n)} = \frac{\frac{\log(n)}{\log(2)}}{\frac{\log(n)}{\log(3)}}$$

4. **Simplify:** Invert and multiply to simplify the complex fraction.

$$\frac{\log(n)}{\log(2)} \cdot \frac{\log(3)}{\log(n)}$$

5. **Cancel common terms:** The $\log(n)$ terms cancel out, leaving a constant value:

$$\frac{\log(3)}{\log(2)}$$

The result

The calculation simplifies to a single constant, which can be approximated using a calculator:

$$\frac{\log_2(n)}{\log_3(n)} = \log_2(3) \approx 1.585$$

revise the calculation rules of log afterwards!!! totally got lose.

4. Extension 4: $\log n \ll (\log n)^2$

log n / (log n)^2 = 1 / log n

When n → ∞, log n → ∞, log n / (log n)^2 = 1 / log n → 0

Therefore, log n << (log n) ^ 2

5. Extension 5: $n \log n \ll n (\log n)^2$

(Just a variation of extension 4 – multiplied n on both sides)

n log n / n (log n)^2 = 1 / log n

When n → ∞, log n → ∞, n log n / n (log n)^2 = 1 / log n → 0

Therefore, n log n << n (log n) ^ 2

Part 2. The Dominant Term Method (video 9)

Having learning how to order the primitive functions by their growth speed, we know that when $n \to \infty$, $1 \ll \log n \ll n \ll n \log n \ll n^2 \ll 2^n \ll n!$.

When these functions are mixed together, we only need to consider the function that grows fastest.

<u>The Dominant Term Method</u>

Suppose we want to compare these two functions

$47n+2n!+17$

$3^n + 102n^3$

We can focus on the dominant term

For $47n+2n!+17$ the dominant term is $n!$
For $3^n + 102n^3$ the dominant term is 3^n

$3^n \ll n!$
So these are not equivalent

Part 3. Big O

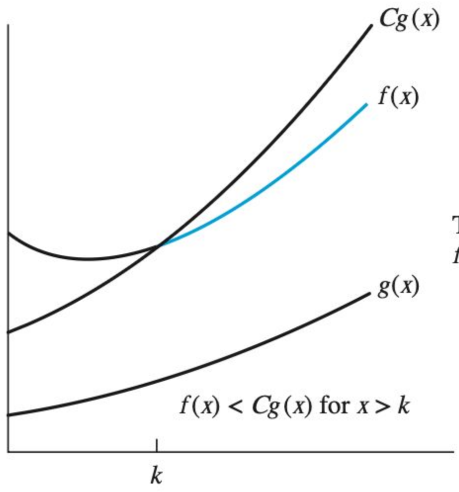
- While we can compare functions asympototically, we can also use another method: big O.
- 1. Definition of Big O (1) (Rosen 3.2, P205):
    - Let f and g be functions from the set of integers or the set of real numbers to the set of real numbers. We say that f (x) is O(g(x)) if there are constants C and k such that
    - $|f(x)| \le C|g(x)|$
    - whenever $x > k$. [This is read as “f (x) is big-oh of g(x).”]
    - Note: C and k are positive real numbers.
- Definition of Big O (2) (video 10):

Big O

If we want to compare two functions f(x) and g(x) we can also use big O.

We say that f(n) is O(g(n)) iff there are positive real numbers c and k such that $0 \le f(n) \le cg(n)$ for every $n \ge k$.

- written form:
    - $f(x) = O(g(x))$
    - Note: However, the equals sign in this notation does n o t represent a genuine equality.
    - or: $f(x) \in O(g(x))$
    - (Rosen 3.2, P207)



The part of the graph of $f(x)$ that satisfies $f(x) < Cg(x)$ is shown in color.

FIGURE 2 **The Function $f(x)$ is $O(g(x))$.**

(Picture: P208)

- What does this mean?
    - Intuitively, the definition that f (x) is O(g(x)) says that **f(x) grows slower** that some fixed multiple of g(x) as x grows without bound. (Rosen 3.2, P205)
    - The application in computer science: big O can help us compare the running time of different algorithms, when n is a very large amount of input, and f(n) and g(n) are the running time (Lecture 12, Q&A).
- C and k: "Witnesses" to the relationship f (x) is O(g(x))
    - To prove big O, we only need to find ONE pair of C and k.
    - But in fact, there are infinitely many pairs of C and k – if C and k can meet the requirements, any C' > C and k' > k can also do. (Rosen 3.2, P 205)
- How to find C and k: **find k first, then find C**
    - A useful approach for finding a pair of witnesses is to first **select a value of k for which the size of |f (x)| can be readily estimated when x > k** and to see whether we can use this estimate to find a value of C for which $|f(x)| \le C|g(x)|$ for x > k. (Rosen 3.2, P206)

- If f(x) is O(g(x)), which mean that there are C and k such that |f (x)| ≤ C|g(x)| (x > k),then g(x) can be replaced by any function with larger values then itself. (Rosen 3.2, P207)

$$|f(x)| \le C|g(x)| \qquad \text{if } x > k,$$

and if $|h(x)| > |g(x)|$ for all $x > k$, then

$$|f(x)| \le C|h(x)| \qquad \text{if } x > k.$$

Hence, $f(x)$ is $O(h(x))$.

- "Same order": f (x) is O(g(x)) and g(x) is O(f (x))
    - e.g. (x+1)^2 and x^2 (Proof: Rosen 3.2 P206-207; Lecture 12)

- How can we prove f(x) is NOT O(g(x))? (Rosen 3.2 P209)

EXAMPLE 3 Show that n^2 is not $O(n)$.

Solution: To show that n^2 is not $O(n)$, we must show that no pair of witnesses C and k exist such that $n^2 \leq Cn$ whenever $n > k$. We will use a proof by contradiction to show this.

Suppose that there are constants C and k for which $n^2 \leq Cn$ whenever $n > k$. Observe that when $n > 0$ we can divide both sides of the inequality $n^2 \leq Cn$ by n to obtain the equivalent inequality $n \leq C$. However, no matter what C and k are, the inequality $n \leq C$ cannot hold for all n with $n > k$. In particular, once we set a value of k, we see that when n is larger than the maximum of k and C, it is not true that $n \leq C$ even though $n > k$. This contradiction shows that n^2 in not $O(n)$. ◀