Lecture 5: Properties of asymptotic notations / Common notations and summations

CS303: Algorithms

Prepared on January 17, 2014

1 Properties of asymptotic notations

1. $f(n) \in O(f(n))$
   Proof: Let $c = 1, n_0 = 1$, then for any $n \geq n_0, f(n) \leq c * f(n)$.
   Similarly, this property holds for Ω and Θ notations.

2. $f(n) \in O(g(n))$ if and only if $g(n) \in \Omega(f(n))$
   This can be proved using the definition. It is the reflective relatiopship between O and Θ.

3. $f(n) \in O(g(n)), g(n) \in O(h(n))$, then $f(n) \in O(h(n))$.
   This is the transitive property. It is also applicable to Θ and Ω.

4. $f_1(n) \in O(g_1(n), f_2(n) \in O(g_2(n))$, then $f_1(n) + f_2(n) \in max(g_1(n), g_2(n))$
   What this formula tells is that if an algorithm has two consecutive parts (one procedure followed by the other one. e.g. sort then search), then the overall algorithm efficiency is determined by the part with a larger order of growth, i.e. its least efficient part.

2 Terminologies ($3.2)

1. Monotonicity
   - $f(n)$ is **monotonically increasing** if $m \leq n \rightarrow f(m) \leq f(n)$
   - $f(n)$ is **monotonically decreasing** if $m \leq n \rightarrow f(m) \geq f(n)$
   - strictly increasing or decreasing basically removes the equal sign in the above definition

2. Exponentials
   - $$\begin{aligned} a^{-1} &= \frac{1}{a} \\ (a^m)^n &= a^{mn} \\ a^m a^n &= a^{(m+n)} \end{aligned}$$
   - As discussed in class, polynomials n^b has a slower order of growth than exponentials a^n where $a > 1$

3. Logarithms

   - Notations

     -
       $$\begin{aligned} lg\ n &= log_2 n \text{ (binary log)} \\ ln\ n &= log_e n \text{ (natural log)} \\ lg^k n &= (lg\ n)^k \\ lg\ n^k &= klg\ n \end{aligned}$$

   - Properties

     - $lg\ n$ is strictly increasing
     -
       $$\begin{aligned} a &= b^{log_b a} \\ lg(ab) &= lg\ a + lg\ b \\ log_b a &= \frac{log_c a}{log_c b} \\ lg\ 1 &= 0 \\ lg\ (1/a) &= -lg\ a \\ log_b a &= \frac{1}{log_a b} \\ a^{log_b c} &= c^{log_b a} \end{aligned}$$

     - log functions grows slower than any polynomials

4. Factorials
   Stirling's approximation $n! = \sqrt{2\pi n}(\frac{n}{e})^n(1 + \Theta(\frac{1}{n}))$.
   Thus we know $lg(n!) = \Theta(nlg\ n)$

3 Commonly used summations (appendix A)

1. Arithmetic series (finite): $\sum_{k=1}^{n} k = 1 + 2 + \ldots n = \frac{1}{2}n(n+1) = \Theta(n^2)$.

2. Geometric series (finite): $\sum_{k=0}^{n} x^k = 1 + x + x^2 \ldots x^n = \frac{x^{n+1} - 1}{x - 1}$ if $x \neq 1$.

3. Geometric series (infinite): $\sum_{k=0}^{\infty} x^k = 1 + x + x^2 + \ldots = \frac{1}{1 - x}$ if $|x| < 1$.

4. Square series (finite): $\sum_{k=0}^{n} x^2 = \frac{n(n+1)(2n+1)}{6} = \Theta(n^3)$.

5. Cube series (finite): $\sum_{k=0}^{n} x^3 = \frac{n^2(n+1)^2}{4} = \Theta(n^4)$.

6. Harmonic number $H_n = \sum_{k=1}^{n} \frac{1}{k} = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n} = ln\ n + O(1)$.

Example:

Find out the efficiency of the following algorithm:

```
Algorithm: Mystery(n)
//Input: a non-negative n
S←0
for i← 1 to n do
  S←S+i*i
return S
```

4 How to compare algorithm efficiency?

If we know the exact $T(n)$ for each algorithm, we can simple compare them. If the $T(n)s$ are too complicated, we can use division with limit to compare their efficiency.

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \begin{cases} 0 & \text{implies that f has a smaller order of growth than g} \\ c > 0 & \text{implies that f and g have the same order of growth} \\ \infty & \text{implies that f has a larger order of growth than g} \end{cases}$$

What's even better is we can use L'Hopital's rule to take derivatives. $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{f'(n)}{g'(n)}$

Example

1. Is it true that 2^n and 3^n have the same order of growth? i.e. $2^n \in \Theta(3^n)$?
   $\lim_{n\to\infty} \frac{2^n}{3^n} = \lim_{n\to\infty} (\frac{2}{3})^n = 0$. Thus $2^n \in O(3^n)$.
2. Is it true that $log_2 n$ and $log_3 n$ have the same order of growth? i.e. $log_2 n \in \Theta(log_3 n)$?
   $\lim_{n\to\infty} \frac{log_2 n}{log_3 n} = \lim_{n\to\infty} (\frac{log_2 e \times \frac{1}{n}}{log_3 e \times \frac{1}{n}}) = const$. Thus $log_2 n \in \Theta(log_3 n)$. So in the future, we can omit the base of logs and use default base as 2.
3. For each of the following functions, indicate the class $\Theta(g(n))$ belongs to using the simplest form possible.
   - $(n^2+1)^{10}$
   - $\sqrt{10n^2 - 7n + 3}$
   - $2nln(n+2)^2 + (n+2)^2 log(\frac{n}{2})$
   - $2^{n+1} + 3^{n-1}$
   - $\lfloor log_2 n \rfloor$

   Answers: $n^{20}, n, n^3 logn, 3^n, logn$