More on Asymptotic Notation

January 23, 2002

- **How to use asymptotic notation for algorithm analysis.**

Asymptotic notation is used to determine rough estimates of relative running time of algorithms. In class we saw an example of *worst-case* analysis of pseudocode that led to a big-O estimate. A worst-case analysis of any algorithm will always yield such an estimate, because it gives an upper bound on the running time $T(n)$ of the algorithm, that is $T(n) \leq g(n)$, and so $T(n) \in O(g(n))$. In some cases, an exact analysis of the running time is possible, and then we get $T(n) = g(n)$, so $T(n) \in \Theta(g(n))$. Here is an example:

```
a <- 0                               1 unit   1 time
for i <- 1 to n do                   1 unit   n times
       for j <- 1 to i do            1 unit   n(n+1)/2 times
              a <- a+1               1 unit   n(n+1)/2 times
```

where the times for the inner loop have been computed as follows: for each i from 1 to n, the loop is executed i times, so the total number of times is

$$1 + 2 + 3 + \ldots + n = \sum_{i=1}^{n} i = n(n+1)/2$$

(see Appendix A.1 in the book for this, and other summation formulas). Hence in this case

$$T(n) = 1 + n + 2n(n+1)/2 = n^2 + 2n + 1.$$

If we write $g(n) = n^2 + 2n + 1$, then $T(n) \in \Theta(g(n))$, that is $T(n) \in \Theta(n^2 + 2n + 1)$. We actually write $T(n) \in \Theta(n^2)$, as recommended by the following rule.

- **Be as simple and as precise as possible in analysis.**

Although the definitions of asymptotic notation allow one to write, for example, $T(n) \in O(3n^2 + 2)$, in practice we never write something like that; we simplify the function in between the parentheses as much as possible (in terms of rate of growth), and write instead $T(n) \in O(n^2)$. Similarly, we don't write, for example, $T(n) \in \Theta(4n^3 - n^2 + 3)$, but $T(n) \in \Theta(n^3)$. Also, it is wrong to write, for instance, $O(\sum_{i=1}^{n} i)$; write instead $O(n^2)$, after computing the sum.

Also, the analysis should be as precise as possible, in the sense that it should provide the closest asymptotic bounds for functions; for example, while it is true that $3 \in O(n^2)$, we know that 3 is actually constant time, rather than quadratic time, so $3 \in O(1)$ or $3 \in \Theta(1)$ is the right analysis.

In the spirit of the simplicity rule above, when we are to compare, for instance, two candidate algorithms A and B having running times $T_A(n) = n^2 - 3n + 4$ and $T_B(n) = 5n^3 + 3$, rather than writing $T_A(n) \in O(T_B(n))$, we write $T_A(n) \in \Theta(n^2)$, and $T_B(n) \in \Theta(n^3)$, and then we conclude that A is better than B, using the fact that n^2 (quadratic) is better than n^3 (cubic) time, since $n^2 \in O(n^3)$. A comparison of the most common functions is given next.

- **Common functions.**

The following relations hold for some of the most common functions encountered in algorithm analysis (where "$\leq$" means "better than"):

$$O(1) \leq O(\log n) \leq O(n) \leq O(n \log n) \leq O(n^2) \leq O(2^n).$$

The same relations hold for Θ instead of O. Also, the following general rules apply:

1. logarithmic is better than polynomial: $O(\log^k n) \leq O(n^l)$, for any constants k, l where $l > 0$;
2. lower-degree polynomial is better than higher-degree polynomial: $O(n^k) \leq O(n^l)$, for any constants k, l with $k < l$;
3. polynomial is better than exponential: $O(n^k) \leq O(l^n)$, for any constants k, l, with $l > 1$.

The same rules apply for Θ instead of O.

- **Other things to prove about asymptotic bounds.**

1. Prove that

$$f(n) \in O(g(n)) \quad \text{iff} \quad g(n) \in \Omega(f(n))$$

. **Proof:** By definition, $f(n) \in O(g(n))$ iff there exist constants $c, n_0 > 0$ such that $0 \leq f(n) \leq cg(n)$, for all $n \geq n_0$. Dividing by c, we get $0 \leq (1/c)f(n) \leq g(n)$, for all $n \geq n_0$. But this is just the definition of $g(n) \in \Omega(f(n))$, in which $c = 1/c$.

2. Prove that $3n^3 \notin O(n^2)$.

Proof: By contradiction. Suppose $3n^3 \in O(n^2)$, Then, by definition, we have constants $c, n_0 > 0$ such that $0 \leq 3n^3 \leq cn^2$, for all $n \geq n_0$. Dividing by n^2 we get $0 \leq 3n \leq c$, for all $n \geq n_0$. This means that $n \leq c/3$, for all $n \geq n_0$. But this is obviously not true for $n > \max\{c/3, n_0\}$.

3. Fill in the most appropriate symbol from $\{O, \Omega, \Theta\}$ at position _ . Justify your answer.

(a) $n \log n^2 \in \_(n \log n)$;

(b) $2^{\log n + n} \in \_(2^n)$.

Solution: (a) Using the property of logarithms $\log a^b = b \log a$, we get $n \log n^2 = 2n \log n$, so $n \log n^2 \in \Theta(n \log n)$.

(b) Using the fact that $2^{\log a} = a$, we get $2^{\log n + n} = 2^{\log n} \cdot 2^n = n2^n$, so $2^{\log n + n} \in \Omega(2^n)$ (since $n2^n \geq 2^n$ for all $n \geq 1$).

Read carefully Chapter 3 of the book, with all the details pertaining to O, Θ, Ω, including Section 3.2. Be prepared to use formulas from this section, and from Appendix A of the book at any time.