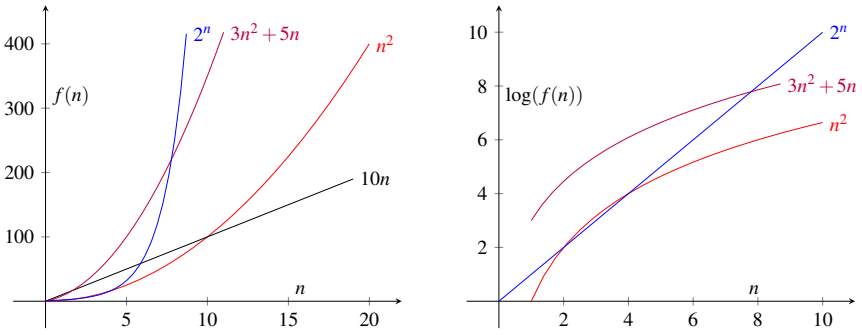
Lecture 1: Asymptotic Notations, Euclid's Algorithm

Scriber: Chenwei Wu

January 10, 2019

1 Asymptotic Notations

The Asymptotic Notations are notations that measures roughly how much time or space an algorithm requires. Thus, we will only keep the most weighted term of a function. E.g., in Asymptotic Notation, n^2 is similar to $3n^2+5n$, but is not similar to 2^n. The intuition can be seen in the two figures below: n^2 and $3n^2+5n$ shares similar speed of growth, while 2^n grows more quickly.



1.1 Definitions

Definition 1. *$f(n) = O(g(n))$, if there exist constants $C > 0$ and $n_0 \geq 0$ such that for every $n \geq n_0$, $f(n) \leq C \cdot g(n)$*

This definition can be roughly considered as "$f(n) \leq g(n)$".

Definition 2. *$f(n) = \Omega(g(n))$, if there exist constants $C > 0$ and $n_0 \geq 0$ such that for every $n \geq n_0$, $f(n) \geq C \cdot g(n)$*

This definition can be roughly considered as "$f(n) \geq g(n)$".

Definition 3. *$f(n) = \Theta(g(n))$, if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$*

This definition can be roughly considered as "$f(n) = g(n)$".

1.2 Examples

(1) $3n^2 + 5n = O(n^2)$

Proof. Let $C = 8$, $n_0 = 1$, then for any $n \geq n_0$,

$$f(n) = 3n^2 + 5n \leq 3n^2 + 5n^2 = 8n^2 = 8 \cdot g(n).$$

□

(2) $2^n \neq O(n^2)$

Proof. First, we claim that when $n \geq 1024$, $n \geq 4\log_2 n$.

Now, for any constant $C > 0$ and $n_0 \geq 0$, choose $n > \max\{1024, 2\log_2 C, n_0\}$, we have

$$n - 2\log_2 n \geq n - \frac{n}{2} = \frac{n}{2} > \log_2 C.$$

(The first inequality holds because $n \geq 4\log_2 n$, and the second inequality holds because $n > 2\log_2 C$.) Thus,

$$n > \log_2 C + 2\log_2 n.$$

Take 2^x on both sides, we get

$$2^n > cn^2.$$

□

(3) There's a useful inequality in asymptotic notations

$$\log n < \sqrt{n} < n < n\log n < n^2 < n^3 < 2^n < 3^n.$$

Note that the log function can take any base, whose reason is in example **(4)**.
(4)$\log_2 n = \Theta(\log_3 n)$

Proof. $\log_2 n = \log_2 3 \cdot \log_3 n$. □

1.3 The Importance of Asymptotic Notations

For this section, let's consider Bubble Sort, which is a well-known sorting algorithm, to take a close look at the usage and importance of asymptotic notations.

1.3.1 Algorithm

Bubble Sort is a sort algorithm which compares each element to its neighbor and swap if not in order. The pseudocode is as below:

```
for i = n-1 downto 1
    for j = 1 to i
        if a[j]>a[j+1] then swap(a[j],a[j+1])
```

1.3.2 Exact number of comparisons for array of length n

Each round, the inner loop will run i times in each outer loop round while i goes from $n-1$ down to 1. Thus,

$$T(n) = (n-1) + (n-2) + ... + 1 = \frac{n(n-1)}{2}.$$

1.3.3 Asymptotic Analysis of number of comparisons

Compare to computing the exact $T(n)$, it is much easier to show $T(n) = \Theta(n^2)$.

Proof. In the summation above, each term is smaller than n, so

$$T(n) \leq n + n + \cdots + n = n(n-1) \leq n^2 \Rightarrow T(n) = O(n^2).$$

(In the right hand side of the first inequality above, there are $(n-1)$ number of ns)
Similarly,

$$T(n) \geq \frac{n}{2} + \frac{n+1}{2} + \cdots + (n-1) \geq \frac{n}{2} + \frac{n}{2} + \cdots + \frac{n}{2} = \frac{n^2}{4} = \Omega(n^2).$$

(In the right hand side of the second inequality above, there are $\frac{n}{2}$ number of $\frac{n}{2}$s) □

Here we are assuming n is even. When n is odd, we can replace $\frac{n}{2}$ with $\frac{n-1}{2}$ and get $\frac{n^2-1}{4}$, which is also $\Omega(n^2)$.

Sometimes the algorithm become more complicated so there's not always a smart way to obtain an accurate result, e.g., if our algorithm calls Bubble Sort on n arrays of size $1, 2, \cdots, n$, then the accurate running time will be hard to compute. However, we can easily prove that the new algorithm runs in $\Theta(n^3)$ time. Thus, by using asymptotic notation, we can obtain a bound for running time more easily and for more occasions.

2 Euclid's Algorithm

The Euclid's Algorithm is an algorithm for computing greatest common divisor (gcd) of 2 positive integers. For example,

$$gcd(15, 9) = 3.$$

2.1 Algorithm

Algorithm 1 Euclid's Algorithm

```
1: if b == 0 then
2:     return a
3: else
4:     return gcd(b, a mod b)
```

Example Run:

$$gcd(15,9) \Rightarrow gcd(9,6) \Rightarrow gcd(6,3) \Rightarrow gcd(3,0) \Rightarrow 3$$

2.2 Proof of Correctness

Proof. We will use induction to prove the correctness of Euclid's algorithm. The induction hypothesis is the following:

Induction Hypothesis: For any $b \leq n$, $gcd(a,b)$ computes the greatest common divisor of a and b correctly.

Base Case:
If $b = 0$, then $gcd(a,0) = a$, which is correct.

Induction:
Suppose the IH is true for $b \leq n$. When $b = n+1$, the algorithm outputs $gcd(b, a \bmod b)$. Since $0 \leq a \bmod b \leq n$, by IH we know Euclid's algorithm computes $gcd(b, a \bmod b)$ correctly.

Therefore, we only need to show $gcd(a,b) = gcd(b, a \bmod b)$. We do that by showing that the set of common divisors for (a,b) and $(b, a \bmod b)$ are the same, which is to say:
(1) If k is a common divisor of (a,b), then k is also a common divisor of $(b, a \bmod b)$.
(2) If k is a common divisor of $(b, a \bmod b)$, then k is also a common divisor of (a,b).

Proof of (1): By definition we know $a \bmod b = a - zb$ for some integer z, so

$$\frac{a \bmod b}{k} = \frac{a - zb}{k} = \frac{a}{k} - z\frac{b}{k}.$$

Since k is a common divisor of (a,b), $\frac{a}{k}$ and $\frac{b}{k}$ are integers. Hence $\frac{a \bmod b}{k} = \frac{a}{k} - z\frac{b}{k}$ is also a integer, which means k divides both b and $a \bmod b$.
Proof of (2): By definition we know $a \bmod b = a - zb$ for some integer z, so

$$\frac{a}{k} = \frac{(a \bmod b) + zb}{k} = \frac{a \bmod b}{k} + z\frac{b}{k}.$$

Since k is a common divisor of $(b, a \bmod b)$, $\frac{b}{k}$ and $\frac{a \bmod b}{k}$ are integers. Hence $\frac{a}{k} = \frac{a \bmod b}{k} + z\frac{b}{k}$ is also an integer, which means k divides both a and b.

In a word, $gcd(a,b) = gcd(b, a \bmod b)$, so by induction, Euclid's algorithm is correct. □