Lecture 1: Asymptotic Notations, Euclid's Algorithm

Scriber: Mo Zhou

January 8, 2020

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. Usually, we use $f(n)$ to represent the running time of an algorithm on an input of size n. 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) There's a useful inequality in asymptotic notations (also see figure 1)

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

(2) $f(n) = 3n^2 + 6n$, then $f(n) = O(n^2)$.

Proof. Let $C = 9$, $n_0 = 1$, then for any $n \geq n_0$, we have $n^2 \geq n$. Thus,

$$f(n) = 3n^2 + 6n \leq 3n^2 + 6n^2 = 9n^2 = 9 \cdot g(n).$$

□

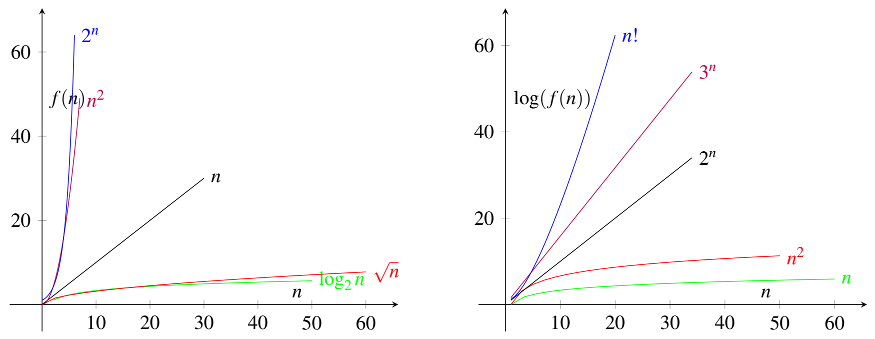

Figure 1: $\log n < \sqrt{n} < n < n\log n < n^2 < n^3 < 2^n < n!$.

(3) $f(n) = n\log_2 n$, then $f(n) \neq O(n)$.

Proof. For every $C > 0$, $n_0 > 0$, we can choose n such that $n \geq n_0$ and $n > 2^C$. This implies that $\log_2 n > C$. For this n, we have

$$n\log_2 n > Cn,$$

which contradicts with the definition of $f(n) = O(n)$. □

1.3 The Benefit of Asymptotic Notations

Consider the following algorithm,

```
for i = 1 to n-1
    for j = i+1 to n
        do something (running time: 1)
```

We first could compute the exact running time. Each round, the inner loop will run $n-i$ times in each outer loop round while i goes from 1 to $n-1$. Thus,

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

Now, let us try to use asymptotic notations to compute $f(n)$. In the summation above, each term is smaller than n, so

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

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

Compare to computing the exact $f(n)$, it is much easier to show $f(n) = O(n^2)$ when using asymptotic notations. Sometimes the algorithm becomes more complicated so there's not always a smart way to obtain an accurate result, e.g., if our algorithm calls some other algorithm as subroutine whose exact running time is difficult to get, then the accurate running time of total algorithm will be hard to compute. However, 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, which is the largest number c that divides both a and b. For example,

$$gcd(12,20) = 4,$$

since $12/4 = 3$ and $20/4 = 5$.

2.1 Algorithm

Algorithm 1 gcd(a,b) (Euclid's Algorithm)

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

Example Run:

$$gcd(12,20) = gcd(20,12) = gcd(12,8) = gcd(8,4) = gcd(4,0) = 4.$$

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 (IH): 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 $b \leq n$. We want to prove IH is also true for $b = n+1$. 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. $\square$