[§4.8] Algorithms

3.1 Introduction & Notation

- An **algorithm** is a list of step-by-step instructions on **how to** accomplish a task. Required properties include:
  - Input/Output -- An algorithm takes some input and produces an output.
  - Precision -- Each step is clearly indicated and unambiguous.
  - Finiteness -- The algorithm stops after a finite number of steps.

  Example: Find the maximum of three numbers, a, b, and c

  - Non-algorithm
    "Compare the three (input) numbers and return (or output) the largest one."
  - Algorithm
    - Step 1: Let x (another number) to have the value of a.
    - Step 2: If b > x, then set x's value to b.
    - Step 3: If c > x, then set x's value to c.
    - Step 4: Return x.
- Algorithms are often written as **pseudocode**, which is sort of the middle ground between a formal programming language and plain English.
- Some of the common elements of pseudocode:
  - Each algorithm starts with "**procedure** name(input values)".
  - There must be at least one "**return** value" statement.
    NOTE: A procedure can only return ONE value only.
  - *Operators* are:
    - +, -, *, /, mod
    - <, <=, >, >=
    - ==, !=
  - The *assignment operator* is := (e.g. "x := 2").
- The basic constructs are
  - "**if** cond **then** action1 **else** action2"
  - "**for** var := initial_value **to** final value **do** action"
  - "**while** cond **do** action"
  - "**repeat** action **until** cond"
  - "**begin** statement$_1$ .. statement$_n$ **end**"
- *Example:* Algorithm *max* which finds the maximum of three numbers, a, b, and c.

  Input: Three numbers a, b, c
  Output: A number which is the largest of the tree input numbers

```
procedure max(a, b, c)
x := a          // x is a local/temporary variable
if b > x then
  x := b        // update x
if c > x then
  x := c        // update x
return x
```

3.2 Example Algorithms

1. Algorithm *find_largest* which finds the largest number in the sequence $s_1,..,s_n$.

   Input: A list of integers $s = \{s_1,..,s_n\}$, and the length n of the list
   Output: A number which is the largest in s.

2. Algorithm *Div* which calculate, for a given integer (n) and a positive integer (d), the quotient (q) and remainder (r) where q and r ate integers. Division is done by repeated subtraction, not by arithmetic division.

   Input: n (an integer), and d (a positive integer)
   Output: q and r (integers)

   ```
   procedure Div(n, d, q, r)
   ```

3. Algorithm *is_even* which tests whether a positive integer m is even

   Input: A positive integer m
   Output: **true** if m is even; **false** if m is odd

3.3 The Euclidean Algorithm

- An algorithm which finds the **greatest common divisor (gcd)** of two integers.

  Example: gcd(30, 105) = 15

divisor of 30	
divisor of 105	
common	

- **"divides"**
  - If a, b and q are integers (where b != 0) satisfying a = bq, we say that "b **divides** a", noted b | a. The value q is called the **quotient**, and b a **divisor** of a.

    Examples: 15 | 30, and the quotient is 2.

- Properties of divisors:

  Let m, n and c be integers.

  1. If c | m and c | n, then c | (m + n)
  2. If c | m and c | n, then c | (m - n)
  3. If c | m, then c | mn

- **<u>Theorem:</u>** For all integers a, b, q and r, such that a = bq + r, and a >= 0, b > 0 and 0 <= r < b, then gcd(a, b) = gcd(b, r).

  First, let's check some cases:

  - 105 = 30 * 3 + 15 -- gcd(105, 30) = gcd(30, 15) = 15
  - 504 = 396 * 1 + 108 -- gcd(504, 396) = gcd(396, 108) = 36

  <u>Proof:</u>

  Let C1 be the set of common divisors of a and b, and C2 be the set of common divisors of b and r. We show C1 ⊇ C2, and C2 ⊇ C1. Then we can conclude C1 = C2.

  1) Show C1 ⊇ C2
  Let c ∈ C1 be a common divisor of a and b, that is, c | a and c | b.
  From the third property of divisors, we get that c | bq.
  Then from the second property of divisors, we get that c | (a - bq) = r. ... (A)
  From (A) and hypothesis, we have that c | b and c | r. Therefore, c is a common divisor of b and r, that is, c ∈ C2.

  2) Show C2 ⊇ C1
  Let c ∈ C2 be a common divisor of b and r, that is, c | b and c | r.
  From the third property of divisors, we get that c | bq.
  Then from the first property of divisors, we get that c | (bq + r) =a. ... (B)
  From (B) and hypothesis, c | a and c | b. Therefore, c is a common divisor of a and b, that is, c ∈ C1.

  From 1) and 2), we showed that all common divisor of a and b are also common divisors of b and r. In particular, this is true for the greatest common divisor, i.e., gcd(a, b) = gcd(b, r).

  Algorithm

```
procedure gcd(a, b)
if a < b, then
  swap(a, b)     // make a the larger of the two
while b != 0 do
  begin
    r := a mod b
    a := b
    b := r
  end
return a
```

iteration		a	b	r
1	gcd(105, 30)	105	30	15
2				
3				

- *Example:* gcd(110, 273) -- Section 3.3, question #2

iteration		a	b	r
1				
2				
3				