Introduction to Algorithms

CIS008-2 Logic and Foundations of Mathematics

David Goodwin

david.goodwin@perisic.com

University of Bedfordshire

12:00, Friday 11th Novemeber 2011

OUTLINE

Characteristics of an Algorithm

Input The algorithm *receives* input.

Output The algorithm *produces* output.

Precision Steps that are precisely stated.

Determinism The intermediate results of each step of execution are unique and determined only by the inputs and the results of the preceding steps.

Finiteness The algorithm terminates; it stops after finitely many instructions have been executed.

Correctness The output produced by the algorithm is correct; the algorithm correctly solves the problem.

Generality The algorithm applies to a set of inputs.

Asignment operator

$=$ denotes the assignment operator. In pseudocode, $x = y$ means "copy the value of y to x" or "replace the current value of x by the value of y".

Example

Suppose that the value of x is 5 and the value of y is 10. After

$$x = y$$

the value of x is 10 and the value of y is 10.

Arithmetic operators

- $+$ is the normal representation for addition
- $-$ is the normal representation for subtraction
- $*$ is a common representation for multiplication
- $/$ is a common representation for division

With arithmetic operations, we must observe the *operational precedence*:

- Multiplication and division always take precedence over addition and subtraction.
- If and addition or subtraction is to be made first, it must be enclosed by parentheses.
- We also note the left to right rule of precedence for multiplication and division.

University of Bedfordshire

Relation operators

- $==$ compare equality
- $\neg =$ compare non-equality
- $<$ compare value to be less than
- $>$ compare value to be greater than
- $\leq$ compare value to be less than or equal to
- $\geq$ compare value to be greater than or equal to

We test some kind of relation between two numbers.

Logical operators

- $\wedge$ conjunction: indicates "and"; a conjunction is true only when both of its components are true
- $\vee$ disjunction: indicates "or"; a disjunction is true when at least one of its components is true
- $\neg$ negation: $\neg p$ reads "not p", "non p", or "negation of p"

Logical operators will be studied in further lectures

IF ELSE STATEMENT

if (*condition*)
    action 1
else
    action 2

If *condition* is true then *action 1* is executed and control passes to the statement following *action 2*. If *condition* is false *action 2* is executed and control passes to the statement following *action 2*. The If statement can be constructed without an Else, in which case If *condition* if false, do nothing and control passes to the statement following *action 1*.

WHILE LOOP

while (*condition*)
  action

If *condition* is true then *action* is executed and this sequence is repeated until *condition* becomes false, then control is passed immediately to the statement following the *action*.

Here we must be carfull not to unintentionally create an infinite loop i.e. if *condition* can never be met, the *action* is repeatedly executed and will not terminate. This is bad programming and bad problem solving i.e. you should have an idea that *condition* will be met at some point through the possible results *action* can give.

For Loop

for *var* = *init* to *limit*
    action

When the for loop is executed, *action* is executed for values of *var* from *init* to *limit* (where *init* and *limit* are integer values).

Function

A function is a unit of code that can recieve input, perform computations, and produce output.

function_name(*parameters separated by commas*){
    code for performing computations
}

Return Statement

return x teminates a function and returns the value of x to the invoker of the function. Without x the return simply terminates the function. If there is no return statement, the function terminates just before the closing brace.

PRIMALITY-TESTING

This algorithm determines whether the integer $n > 1$ is prime. If n is prime the algorithm returns 0. If n is composite, the algorithm returns a divisor d satisfying $2 \geq d \geq \sqrt{n}$. To test whether d divides n, the algorithm checks whether $n \mod (d)$ is zero.

Primality-Testing Algorithm

Input: n
Output: d

Primality-Testing Algorithm

Input: n
Output: d

$is\_prime(n)\{$
  if $(n \mod d == 0)$
    return d
$\}$

Primality-Testing Algorithm

Input: n
Output: d

```
is_prime(n){
    for d = 2 to ⌊√n⌋
        if (n mod d == 0)
            return d
    return 0
}
```

Example of Euclid's Algorithm

Example (Euclid's Algorithm)

Calculate $gcd(1485, 1745)$ using Euclid's algorithm.
If $a = qb + r$ then $gcd\,(a, b) = gcd\,(b, r)$. We use the equation $a = qb + r$ to find r, then to repeat using $gcd\,(b, r)$. Remember the constraints $\{q \mid q \in \mathbb{Z}\}$ and $\{r \mid r \in \mathbb{Z}$ and $r < b\}$.

$$\begin{aligned}
1745 &= 1485q + r & q &= 1 & r &= 260\\
1485 &= 260q + r & q &= 5 & r &= 185\\
260 &= 185q + r & q &= 1 & r &= 75\\
185 &= 75q + r & q &= 2 & r &= 35\\
75 &= 35q + r & q &= 2 & r &= 5\\
35 &= 5q + r & q &= 7 & r &= 0
\end{aligned}$$

Therefore $gcd\,(1485, 1745) = 5$

Euclid's Algorithm

This algorithm finds the greatest common divisor of the non-negative integers a and b, where a and b are not both zero.

Euclid's Algorithm

Input: a and b // the non-negative integers, not both zero
Output: a // greatest common divisor of a and b

Euclid's Algorithm

Input: a and b // the non-negative integers, not both zero
Output: a // greatest common divisor of a and b

$gcd(a, b)\{$
  while $(b \neg = 0)\{$
    $r = a \mod (b)$
    $a = b$
    $b = r$
  $\}$
  return a
$\}$

Euclid's Algorithm

Input: a and b // the non-negative integers, not both zero
Output: a // greatest common divisor of a and b

```
gcd(a, b){
    // make a the largest of the two inputs
    if (a < b)
        swap(a, b)
    while (b¬ = 0){
        r = a mod (b)
        a = b
        b = r
    }
    return a
}
```

University of Bedfordshire

Example of Bezout's Identity

Example (Bezout's Identity)

Express $gcd(1485, 1745)$ in the form $1485u + 1745v$.
From the previous example we found $gcd(1485, 1745) = 5$

$$
\begin{aligned}
5 &= 75 - (2 \times 35) \\
&= 75 - 2 \times (185 - (2 \times 75)) \\
&= (5 \times 75) - (2 \times 185) \\
&= 5 \times (260 - (1 \times 185)) - (2 \times 185) \\
&= (5 \times 260) - (7 \times 185) \\
&= (5 \times 260) - 7 \times (1485 - (5 \times 260)) \\
&= (40 \times 260) - (7 \times 1485) \\
&= 40 \times (1745 - (1 \times 1485)) - (7 \times 1485) \\
&= (40 \times 1745) - (47 \times 1485) = 69800 - 69795 = 5
\end{aligned}
$$

Bezout's Identity algorithm

This algorithm computes s and t satisfying $gcd(a, b) = sa + tb$, where a and b are non-negative integers, not both zero.

Recursive Euclidean Algorithm

Input: a and b // the non-negative integers, not both zero
Output: a // greatest common divisor of a and b

$gcdr(a, b)\{$
  // make a the largest of the two inputs
  if $(a < b)$
    $swap(a, b)$
  if $(b == 0)$
    return a
  $r = a \mod (b)$
  return $gcdr(b, r)$
$\}$

Bezout's Identity algorithm

Input: a and b // the non-negative integers, not both zero
Output: $gcd(a, b)$ // greatest common divisor of a and b returned
s, t // parameters are stored

```
STgcdr(a, b, s, t){
  if (a < b)
    swap(a, b)
  if (b == 0){
    s = 1
    t = 0
    return a
  }
  q = ⌊a/b⌋
  r = a mod (b)
  g = STgcdr(b, r, s′, t′)
  s = t′
  t = s′ − t′ ∗ q
  return g
}
```

University of Bedfordshire

Class Exercises

Write a pseudocode for an algorithms that determines to following:

1. Swap the values a and b.
2. Determine whether an integer is even or odd without using modulus, floor or ceil.
3. Determine the largest and smallest value of the different values a, b, and c, and output the largest and smallest values.
4. Determine the product of $s_1, s_2, \ldots, s_n$
5. Explicitly determine the modulus function.
6. Explicitly determine the floor and ceil functions.
7. Determine the factorial of an integer n.

Swap the values a and b.

Input: a,b
Output: a,b

$t = a$
$a = b$
$b = t$

Determine whether an integer is even or odd without using modulus, floor or ceil.

Input: i
Output: out

$k = 0$
while $out \geq 0$
    $k = k + 1$
    $out = i - 2 * k$
return out

Determine the largest and smallest value of the different values a, b, and c, and output the largest and smallest values.

Input: a, b, c
Output: l, s

$l = a$
$s = a$
if $b > l$
    $l = b$
else
    $s = b$
if $c > l$
    $l = c$
else
    if $c < s$
        $s = c$