Institut for Matematik og Datalogi
Syddansk Universitet

July 22, 2015
JFB

Introduction to Computer Science E15 – Discussion Sections – Week 36

1. Trace the execution of the following pseudocode on inputs $M = 5$ and $N = 14$. What values do the variables M, N, and Z get during the execution? What is the result? More generally, what does the algorithm calculate?

   (For information on pseudocode see the appendix in *Discrete Mathematics and Its Applications*, by Kenneth H. Rosen. In addition to what is explained there, I use functions, such as CALC and GCD below. They can be called, using an assignment such as `c := CALC(5,14)`, for example. The statement `return(`Z`)` assigns the value variable Z has to the function call. Your answer as to what the algorithm below calculates would be assigned to the variable c if the function was called with `c := CALC(5,14)`.)

```
**CALC**($M, N$):
{ Input: two positive integers $M, N$ }
{ Output: CALC($M, N$) }

    $Z := 0$
    **while** $M \neq 0$
    **begin**
        $Z := Z + N$
        $M := M - 1$
    **end**
    return($Z$)
```

2. Recall that the greatest common divisor algorithm can be written in pseudocode as follows:

```
GCD($M, N$):
{ Input: two positive integers $M, N$ }
{ Output: gcd($M, N$) }

    $A :=$ max($M, N$)
    $B :=$ min($M, N$)

    $Q := A$ div $B$
    $R := A - (Q \cdot B)$
    while $R \neq 0$
    begin
        $A := B$
        $B := R$
        $Q := A$ div $B$
        $R := A - (Q \cdot B)$
    end
    return($B$)
```

The operation "div" is an integer division, so A div B gives the integer result $\lfloor \frac{A}{B} \rfloor$ and $A - (Q \cdot B)$ then gives the remainder of dividing A by B.

- What is the largest number of times you can make the algorithm execute when $M = 34$ and $N \leq 34$? As a challenge, try to generalize this to determine which pairs of integers are "bad" for the greatest common divisor algorithm, where a pair is bad if the algorithms must perform a lot of steps relative to how large the numbers are. (One expects more steps for larger numbers.)
- Using pseudocode as in the appendix of Rosen's textbook (which you are using for your Discrete Methods for Computer Science course), remove the max and min functions and use an **if-then-else** structure instead, along with the symbol $>$ to test if one value is larger than another or not. Your new pseudocode should produce the same result in all cases.
- Using pseudocode represent this algorithm in another way by removing the **while** structure and using a **repeat-until** structure instead (this structure is described in section 5.4 of the textbook

for this course). Your new pseudocode should produce the same result in all cases.

3. Using pseudocode, write up an algorithm for calculating

$$N! = N \cdot (N-1) \cdots 2 \cdot 1$$

(N factorial).

4. Do problem 4 on page 38 of the textbook.

5. Consider the Boolean formula $(a \wedge b \wedge \bar{c}) \vee (\bar{a} \wedge b \wedge c)$. Write a truth table for it Draw a circuit for it. Try to write a smaller formula and circuit for it. (Note that $\bar{x}$ is the same as $\neg x$.)

   (Note that this formula is in what is called Disjunctive Normal Form (DNF). The parts inside parentheses are called clauses. Each clause consists of some variables and/or complements of variables (literals), combined with ANDs, and the clauses are combined with ORs.)

6. Design a circuit, using only AND, OR and NOT gates which takes three bits as input and outputs a 1 if the input has at least two zeros, and a 0 otherwise.

7. Design and draw a circuit containing only AND and XOR gates (with at most two inputs) which takes three bits as input and outputs a 1 if the input has at least two ones, and a 0 otherwise.

   (In the student resources for the course textbook, under the Activities for Chapter 1, there is a simulator for logic circuits which you could use to check your circuit. It is time consuming to use, though.) As an extra challenge, try to do it so that there is only one AND gate, though more XOR gates. (Minimizing the number of AND gates can be useful in some cryptographic applications.)