Institut for Matematik og Datalogi
Syddansk Universitet

March 1, 2002
JFB

# Discrete Mathematics with Applications
# F02 – Lecture 5

## Lecture, February 25

The sections of the DM11 notes on structural induction and the Extended Euclidean algorithm were covered, along with section 3.3 of chapter 3. We also covered sections 2.3 and 2.4 of chapter 2, and through page 139 of section 2.5.

## Lecture, March 4

We will finish section 2.5, along with Lamé's Theorem from chapter 3.

## Lecture, March 11

Klaus Meer will begin on chapter 4.

## Problems to be discussed on March 12

For checking your results on some of the following problems, the following Maple functions should be useful: `igcdex` (extended Euclidean algorithm for integers), `mod` (where the operation `&^` should be used for more efficient modular exponentiation - try them both to compare), `msolve` (solve equations modulo $m$), and `chrem` (Chinese Remainder Algorithm).

- Use the Extended Euclidean Algortihm to compute the following multiplicative inverses:

  1. $17^{-1} \pmod{101}$
  2. $357^{-1} \pmod{1234}$

3. $3125^{-1} \pmod{9987}$

- Solve the following system of congruences:

$$\begin{aligned} x &\equiv 12 \pmod{25} \\ x &\equiv 9 \pmod{26} \\ x &\equiv 23 \pmod{27} \end{aligned}$$

- Solve the following system of congruences (use the Extended Euclidean algorithm, and then apply the Chinese Remainder Theorem):

$$\begin{aligned} 13x &\equiv 4 \pmod{99} \\ 15x &\equiv 56 \pmod{101} \end{aligned}$$

- Problems (40), (57) from section 3.2.

- Problems (2b), 2c, 9, 10, 15, 18, 19, 20, 23, and 35 from section 2.5.

- Exercise 14 on pages 163–164, if I don't do it in the lecture.

The following exercises are designed to give you a feel for how hard factoring is compared to multiplication, or finding primes:

- Use *help* in Maple to find out about the function *ithprime*. Experiment to find out approximately how big a prime it can find. How large is it? Multiply two of the large primes it finds together, and try to factor the result, using the function *ifactor*. Notice how quickly the factors are found for these small numbers. (Large numbers are clearly necessary for security.)

- Finding larger primes.

  In order to find good prime factors $p$ and $q$ for use in RSA, one can choose random numbers of the required length and check each one for primality until finding a prime.

  Maple contains a function *isprime* which will test for primality. Try it on some some small numbers, such as 3, 4, 7, 10. Maple has another function *rand* which returns a random 12-digit number. Try checking three random 12-digit numbers for primality. Try factoring them too. How long do these operations take? Try producing random 12-digit numbers until you find a prime. You can use a *while loop*. You want to continue creating new random numbers until you get a prime, so you can type `while (not isprime(x)) do`, ignore the warning, and type `x:=rand();` on one line and `end do;` on the next. (Remember to initialize $x$ to something not prime.) How many different values were chosen before a prime was found? (I got 32, but you could get another number.) Now create a second prime called $y$ (remember that $y$ will need some value before your start your *while loop*). Multiply $x$ and $y$ together and try factoring the result. This should also go relatively quickly.

- Even larger numbers. Use `rand()` to find even larger primes, multiply them together and try factoring them. How large do the primes have to get before it takes a long time to factor them? Can you find primes which are much longer? Investigate this to see for different length numbers how long it takes to find a prime and how long it takes to factor a number of that length. Does it make any difference if the numbers to be factored have small prime factors?