Discrete logarithm

Exercises

Application of discrete models

Mathematical questions

1. Calculate the following modular powers:
a) $2^{10} \bmod 3$; b) $3^7 \bmod 5$; c) $4^{123} \bmod 5$; d) $5^5 \bmod 7$.

2. Convert a^{19} to the form used by the fast exponentiation.

3. Define the order of a number modulo m. What is the order of 3 modulo 4?

4. Write down Fermat's little theorem. Give an example with a modulus between 10 and 15.

5. Write down Lagrange's theorem. Give an example with a modulus between 10 and 15.

6. Define primitive roots. Is 2 a primitive root modulo 3?

7. Define discrete logarithm. What is $\operatorname{ind}_3 4 \pmod 5$?

8. How do the two participants of the Diffie–Hellman key exchange calculate the common secret?

9. Which mathematical problem needs to be solved to break the Diffie–Hellman key exchange?

Programming exercises

10. Write a function that calculates $a^n \bmod m$

a) using repeated multiplication, with a *single* remainder calculation at the end;

b) using repeated multiplication, with remainder calculations *in each step*;

c) using fast exponentiation.

Test the result using the built-in function `power_mod()`.

11. For $a = 6$ and $m = 11$, find an exponent n for which

a) part a) of the previous exercise is slow, but b) and c) are fast;

b) parts a) and b) of the previous exercise are slow, but c) is fast.

12. Write a function that calculates the order of a given $a \in \mathbb{Z}$ modulo p. Test it for the first few prime numbers p, and for each possible remainder a. In each case, check that the power satisfies the definition of the order, and that the result satisfies Langrange's theorem.

13. Calculate the discrete logarithm of a to the base g modulo p

a) using linear search; b) using the baby-step giant-step algorithm.

Test it for various prime numbers p and primitive roots g, for precalculated powers. Run them also for $p = 12345678923$, $g = 2$ and $a = 9718522932$. Do they run for the same p and g, but for a random a between 1 and $p-1$?

14. Simulate the Diffie–Hellman key exchange, i.e. write a program that executes both participants' steps, calculates the common secret in both ways, and checks if they are indeed equal. Try it for $p = 23$ and $g = 5$, and then for $p = 12345678923$ and $g = 5$.

15. a) Implement the following pseudorandom number generator:

$$x_n := 48271^n x_0 \bmod 2147483647,$$

where x_0 is the seed, i.e. an arbitrary positive integer. Write a function that generates the next number (x_n) from only the previous value (x_{n-1}), i.e. without knowing x_0 and n.

b) How "random" are the generated numbers? Ideally, each bit of the generated numbers would be 1 or 0 with 50%-50% probability. Generate 1000 random numbers (from x_1 to x_{1000}), and create statistics about the bits of the numbers (i.e. their digits in binary), e.g.:

```
bit 0:  500 (0), 500 (1)
bit 1:  491 (0), 509 (1)
[...]
bit 30: 511 (0), 489 (1)
```

(Smaller numbers with less than 31 bits should be treated as if the missing bits were 0.)