Tutorial 1

Exercise 1: Try to formulate what is an (algorithmic) ***problem***, and in particular what is a ***decision*** (or YES/NO problem) and what is a ***optimization*** problem.

Try to specify the following problems as precisely as possible. For each these problems give examples of input instances and the corresponding outputs, and in the case of decision problems give examples of instances where the answer is YES, and examples of instances where the answer is NO.

a) Primality test.
b) Factorization of a natural number into primes.
c) Matrix multiplication.
d) Finding the minimal spanning tree.
e) Longest common subsequence of two strings.
f) Equivalence of deterministic finite automata.
g) Equivalence of non-deterministic finite automata.
h) Determining whether a given context-free grammar generates a nonempty language.
i) Equivalence of context-free grammars.
j) Determining whether a given context-free grammar is unambiguous.

For those above mentioned problems, which are not decision problems, try to think how these problems could be reformulated as decision problems.

Give a definition of an ***algorithmically solvable*** problem and an (algorithmically) ***decidable*** problem.

Try to determine, which of the problems given above are decidable or algorithmically solvable, and which are not. For those problems where you think they are decidable or algorithmically solvable, describe an algorithm that solves them.

Exercise 2: Represent the following algorithm written in a form of pseudocode as a control-flow graph:

Algorithm 1: Insertion sort

```
INSERTION-SORT (A, n):
    for j := 1 to n − 1 do
        x := A[j]
        i := j − 1
        while i ≥ 0 and A[i] > x do
            A[i + 1] := A[i]
            i := i − 1
        A[i + 1] := x
```

Determine number of steps performed by this algorithm when it obtains the pair of values $A = [5, 2, 1, 4, 3]$ and $n = 5$ as an input. (One edge of the control-flow graph corresponds to one instruction, and so to one step of the algorithm.)

Exercise 3: Determine what the following two fragments of programs for Random Access Machine do.

Remark: Symbolic labels are used here instead of explicit addresses of instructions.

```
          R0 := READ()
          R1 := 2
          goto entry
loop:     R0 := R0 − 1
          R1 := R1 ∗ R1
entry:    if (R0 > 0) goto loop
          WRITE(R1)
          halt
```

```
          R3 := R0 + R1
          R3 := R3 − 1
          R2 := [R3]
          goto entry
loop:     R4 := [R3]
          if (R4 ≤ R2) goto entry
          R2 := R4
entry:    R3 := R3 − 1
          if (R3 ≥ R0) goto loop
          WRITE(R2)
          halt
```

Exercise 4: Construct a program for a RAM that reads two numbers x and k from the input and writes the value of k-th bit of the number x (i.e., 0 or 1) on the output. The bits are numbered starting from 0 and 0-th bit is the least significant bit. You can assume that $x \geq 0$ and $k \geq 0$ (i.e., you don't have to consider the cases when $x < 0$ or $k < 0$).

Exercise 5: Consider a variant of RAM whose only arithmetic instructions are addition and subtraction, and addition to that it has an instruction for binary shift right by one bit (i.e., an instruction of the form $R_i := \lfloor R_j/2 \rfloor$). So in this variant, the instruction for multiplication and division are not available. (Other instructions — memory accesses, jumps, branching, etc., are the same as in the definition of RAM presented in the lecture.)

Show that the bahavious of an instruction for multiplication can be simulated by this variant of RAM. For simplicity, you can assume that the multiplied values are nonnegative.

Construct a program for the variant of RAM described above that reads a pair of numbers x and y (where $x \geq 0$ and $y \geq 0$), and writes their product $x \cdot y$ to the output.

a) What is the time complexity of your algorithm, i.e., what be be the total number of executed instructions performed by your program depending on the number of bits used in the representation of numbers x and y?

b) Contruct a program whose time complexity is polynomial with respect to the number of bits of numbers x and y.

c) Try to propose a variant of the program that will not use the intruction for the binary shift right and still be polynomial.

Exercise 6: Propose some way how a Turing machine with some general tape alphabet Γ can be simulated by a Turing machine with tape alphabet $\{0, 1, \square\}$.

How the number of steps is changed in this simulation?

Exercise 7: Propose a way how a Turing machine with two tapes can be simulated by a Turing machine with one tape.

How the number of steps performed by the original two-tape machine is related to the number of steps performed by the constructed one-tape machine?

Generalize your approach to the cases of machines with arbitrary (but finite) number of tapes — i.e., for machines with three tapes, four tapes, etc.

Exercise 8: Propose and describe in detail how a RAM could efficiently simulate a Turing machine:

a) At first, consider a Turing machine with one tape, which is infinite only in one direction.

b) Consider a Turing machine with one tape, which is infinite in both directions.

c) Consider a Turing machine with multiple tapes and with multiple heads on each tape where these tapes can be infinite in both directions.