ADS 1 — exercise sheet 1

Random Access Machine (RAM)

Memory of RAM is a sequence of integers indexed by integers. Program for RAM consists of the following instructions:

1. arithmetic instructions
   $\alpha \leftarrow \beta$ op γ where op is one of $+, -, \times, /,$ mod ,
2. logical instructions
   $\alpha \leftarrow \beta$ op γ where op is one of and, or, xor, <<, >>,
3. control flow instructions
   `halt`
   `goto label`
   `if` α op β `then goto label`
   where op is one of $=, \neq, \leq, \geq, <, >$.

Whre operands α, β, γ can be one of the following:

1. literal constant,
2. dirrectly addressed memory cell written as [const], or,
3. Indirectly addressed memory cell written as [[const]].

If operand appears on the left hand side of $\leftarrow$ it can not be literal constant.

Time of the execution of a program for a given input is the number of executed instructions. Space of the execution is the difference of the maximal and minimal address of a memory cell used by the program.

Excercises

Exercise 1: Implement basic RAM program for primality testing (by considering all possible divisors up to the square root of an input). What is the time and space complexity of this algorithm?

Exercise 2: Implement sieve of Eratosthenes in RAM. What is time and space complexity?

Exercise 3: Consider RAM with unbounded memory. Show how to encode arbitrary large sequence of numbers $c_1, \ldots, c_n$ into single integer C such that individual numbers c_i can be effectively decoded?

Exercise 4: Describe method converting every program in RAM to program which uses only constantly many memory cells (containing possibly very large integer). Program can become arbitrarily slower. How many memory cells are needed?

Exercise 5: Implement algorithm in RAM for testing that given number is a power of two in constant time.

Exercise 6: Describe the difference between DFS as introduced on 2nd lecture over modified BFS with use of stack instead of queue. Describe its implementation using RAM (that does not feature call and return instructions to implement recursive functions).

Exercise 7: Lets have skyscraper with n levels and k eggs. Egg is an object for which there exists a constant m so if we throw the egg away from a window in level at most m, it will never break. If we throw it from any higher level it will always break. Design algorithm to determine value m when number of steps is measured as number of times the egg is thrown out from the window (we have unlimited computation power otherwise) for the following cases:

1. $k = 1$,
2. $k = \infty$,
3. $k = 2$.