In the lecture last Tuesday I learned about algorithms. Apparently an algorithm is a finite sequence of steps for solving a problem. Due to the fact that our work is focused on the computer we might add to this definition that the sequence of steps can be done by a computer.

There are different ways to write algorithms. They can be described using natural language though that is not a good way, since natural language is ambiguous. Further possibilities are a Flow Chart and the Nassi-Shneiderman Diagramm. Both of those possibilities are fine but not perfect, e.g. the Flow Chart gets messy if the particular algorithm is to complex (it will look like a “Spaghetti Diagram”).

Describing an algorithm with pseudo-code is most likely the best solution, since that code can be understood easily and transferred into code of a lot of programming languages without major problems. The termination of an algorithm is an important property of it.

To find out how long it takes the computer to solve/run an algorithm the Random Access Machine can be used. This RAM has register and an infinite number of storage. The commands used are similar to Assembler. Given that, the RAM is able to count instructions and storage cells.

When taking a look at the complexity of and Algorithm (the Big OH) the absolute number is not important but the growth rate. That means, it is important to see, how the operating time changes when the number of runs changes. There can be huge growth rates, depending on the particular algorithm. Thus it is essential to first optimize the algorithm and not the code.