INF 4130 Exercise set 1

Problem 1 (Coding of instances)

As described in the lecture, we model real word problems mathematically to study or solve them. When we want to write down, or code, a problem instance so that it can be given as input to an algorithm (computer program, Turing machine), or presented to another person, we need to decide on a suitable format. We describe the problem instance as a string.

a) Numbers can be written in a variety of alphabets. Compare the approximate (big-O) lengths of natural number codes in decimal, binary and unary alphabet.

(Important notions: Coding, string lengths, exponential vs. linear difference.)

Problem 2 (HAMILTONICITY)

Recall that a formal language is the set of YES-instances for the corresponding (decision) problem. We can write, or code, all instances of a problem with a suitable format that we choose.

a) Show how HAMILTONICITY can be represented as a formal language over the alphabet {0, 1}. Discuss the relationship between the length of the representation and the number of nodes and edges in the input graph.

(Important notions: strings as a formalization of problem instances, formal languages as a formalization of problems. Relationship between the code lengths and the number of elements in the input.)

Problem 3 (Turing machines)

We use the Turing machine as our definition of an algorithm. It is a very simple model, but the simplicity makes it easy to define the notion of running time / time complexity.

a) Construct a Turing machine which recognizes the language over the alphabet {0, 1} which consists of the strings of one or more 0's only. What is the time complexity of your Turing machine?

b) Show how the above construction can be modified into a machine which recognizes the language $L = \{0^k1^k \mid k = 0, 1, 2, ...\}$ (words in L consist of k zeros followed by k ones). How has the time complexity of the machine changed?

(Important notions: Turing machine as a formalization of "algorithm" and "solution". The language recognized by a Turing machine. The complexity of a Turing machine.)

Problem 4 (Universal Turing Machines) [Difficult]

There is an obvious difference between a Turing machine and an ordinary computer: A standard Turing machine executes a single, hard coded, algorithm, while an ordinary computer takes in both an algorithm (program) and input data and runs the program on the input data.

a) Show that there is a Turing machine called Universal Turing machine (UTM) which takes a TM code M and a string I as input and then does the same as the machine M would when started on input I.

(Important notion: UTM)