Approximation Algorithms, exercises week 1

October 11, 2013

difficult problems (in the complexity theoretic sense)

a) Do there exist decision problems for which there does not exist a Turing machine that computes them? If so, can you construct one?

b) Does every problem in NP have an algorithm which decides it deterministically? If not, can you construct such a problem? If so, how much time would it take at most?

nondeterminism

A nondeterministic Turing machine may, every time it proceeds to the next step in its calculation, flip a coin and let its next step depend on the outcome of that coin flip. So the output of a nondeterministic Turing machine is not only determined by its input, but also by the coin flips it makes during its computation.

One could define the class NP as the set of decision problems $A \subseteq \Sigma^*$ for which there exists a nondeterministic Turing machine M which works in polynomial time and $x \in A$ if and only if there is a possibility that M gives output 1 on input x.

Is this definition equivalent to the one given in the lecture, i.e., does it define the same class NP?

changing the restriction on the length of the guess

Let P be the class of decision problems that are decidable in polynomial time and let C be the class of decision problems $A \subseteq \Sigma^*$ for which there exists $V \in$ P and a polynomial p such that

$$A = \{x \in \Sigma^* : \exists y \in \Sigma^* \, (|y| \leq p(|y|) \wedge (x, y) \in V)\}.$$

The difference between the definition the class NP and the definition of the class C is that in the definition of NP $p(|x|)$ is used as restriction on the length of y whereas in the definition of C the length of y is restricted by $p(|y|)$. What can you say about the class C? Is $C \subseteq NP$? Is $NP \subseteq C$?