Remembering some intuitions about NP and NP-completeness

(for more formal definitions and details, see the slides of the EDA course on this same website)

Decision problems and complexity classes

Here we focus on decision problems, the ones with output "yes" or "no", and on *classifying problems* (not algorithms!) according to the time needed to solve them (with the best of the available algorithms), and we will call problem A *harder* than problem B if solving A needs more time than solving B.

For example, given a sequence of integers, the problem of deciding whether it contains the integer 7 can be solved in *linear* time. We say that it belongs to the *class* of problems solvable in linear time. If moreover the input sequence is *ordered*, then we can say more: it belongs to a proper subclass of the problems solvable in linear time, namely the ones solvable in *logarithmic* time (in this case, by binary search). Here we see that in fact what matters is *how fast the running time grows depending on the size of the input*.

Other problems are not linear, but harder. The class of *polynomial* problems is called P. Note that all logarithmic, linear, quadratic, cubic, etc., problems are in P.

Some other problems are even harder, and are not in P. The class of *exponential* problems is called EXP (their running time has the input size n in the exponent; note that for large enough n, the number 2^n is much larger than n^2, n^3, or n^k for whatever constant k). It is known that P $\subset$ EXP (there are problems in EXP that are not in P, such as "generalized chess").

The class NP, membership in NP, NP-hardness and NP completeness

There is a special class, NP, for which it is known that P $\subseteq$ NP $\subseteq$ EXP. NP is the class of problems having a Nondeterministic Polynomial algorithm. Roughly, this means that a problem A is in NP if, whenever the answer to A for a given input is "yes", there is a "witness" that allows one to verify this "yes" in polynomial time.

The most famous problem in NP is SAT, the problem of deciding whether a given propositional input formula F is satisfiable or not. This problem is clearly in NP: if the answer is "yes", the witness is the model, which can be checked in polynomial (even linear) time. Another example of problem in NP is *3-colorability*: can we color each node of a given graph G with one of three colors, such that adjacent nodes get different colors? Here the witness is the coloring, indicating each node's color.

SAT is *NP-hard*: *any* problem in NP can be polynomially *reduced to* (or solved by, or expressed as) a SAT problem. This means that for any problem A in NP and input data D for A, we can build in polynomial time a SAT formula F_D that is satisfiable if, and only if, the answer to A on input D is "yes". Moreover, from a satisfiablity witness of F_D (i.e., a model), it is usually easy to reconstruct a witness (or a "solution") for A on input D.

For example, we can reduce 3-colorability to SAT. Let G be a graph with n nodes. Introducing $3n$ propositional symbols x_{ic} meaning "node i gets color c", let F_G state, for each node i, that it gets at least one color (a clause $x_{i1} \vee x_{i2} \vee x_{i3}$) and, for each edge (i, j), that i and j do not get the same color (three clauses per edge: $\neg x_{i1} \vee \neg x_{j1}$, $\neg x_{i2} \vee \neg x_{j2}$, and $\neg x_{i3} \vee \neg x_{j3}$). Then F_G is satisfiable iff G is 3-colorable, and from any model for F_G it is trivial to reconstruct a 3-coloring for G.

Apart from SAT, many other problems in NP have been proved NP-hard too. Note that, by such reductions, if we had a polynomial algorithm for for *any* single NP-hard problem, then we would have it for *all* problems in NP, that is, we would have P=NP. That would have dramatic consequences, because there are many very important real-world problems in NP. In fact, there is a million-dollar prize (search "millenium problems") for whoever proves either P=NP or P $\neq$ NP.

Since P $\subset$ EXP, at least one of the two inclusions in P $\subseteq$ NP $\subseteq$ EXP is strict, and it is believed that both are, i.e., P $\subset$ NP $\subset$ EXP.

A problem is called *NP-complete* if A) it is in NP and B) it is NP-hard.