Ñ̃o Title

March 6, 2014

1 NP Complete

Decision problem : Any problem with an answer, YES or NO.

P : A decision problem that can be solved in polynomial time. That is, given an instance of the problem, the answer yes or no can be decided in polynomial time.

Example : Given a graph connected G, can its vertices be colored using two colors so that no edge is monochromatic. Algorithm: start with an arbitrary vertex, color it red and all of its neighbors blue and continue. Stop when you run our of vertices or you are forced to make an edge have both of its endpoints be the same color.

NP : A decision problem where instances of the problem for which the answer is yes have proofs that can be verified in polynomial time. This means that if someone gives us an instance of the problem and a certificate (sometimes called a witness) to the answer being yes, we can check that it is correct in polynomial time.

Example: Integer factorization is NP. This is the problem that given integers n and m, is there an integer f with 1 ¡ f ¡ m such that f divides n (f is a small factor of n)? This is a decision problem because the answers are yes or no. If someone hands us an instance of the problem (so they hand us integers n and m) and an integer f with 1 ¡ f ¡ m and claim that f is a factor of n(the certificate) we can check the answer in polynomial time by performing the division n / f.

NP-complete : An NP problem X for which it is possible to reduce any other NP problem Y to X in polynomial time. Intuitively this means that we can solve Y quickly if we know how to solve X quickly. Precisely, Y is reducible to X if there is a polynomial time algorithm f to transform instances x of X to instances y = f(x) of Y in polynomial time with the property that the answer to x is yes if and only if the answer to f(x) is yes.

Example: 3-SAT. This is the problem wherein we are given a conjunction of 3-clause disjunctions (i.e., statements of the form

(x_v11 or x_v21 or x_v31) and (x_v12 or x_v22 or x_v32) and ... and (x_v1n or x_v2n or x_v3n) where each x_vij is a boolean variable or the negation of a variable from a finite predefined list (x_1, x_2, ... x_n). It can be shown that every NP problem can be reduced to 3-SAT. The proof of this is technical and requires use of the technical definition of NP (based on non-deterministic Turing machines and the like). This is known as Cook's theorem.

What makes NP-complete problems important is that if a deterministic polynomial time algorithm can be found to solve one of them, every NP problem is solvable in polynomial time (one problem to rule them all).

NP-hard : Intuitively these are the problems that are even harder than the NP-complete problems. Note that NP-hard problems do not have to be in NP (they do not have to be decision problems). The precise definition here is that a problem X is NP-hard if there is an NP-complete problem Y such that Y is reducible to X in polynomial time. But since any NP-complete problem can be reduced to any other NP-complete problem in polynomial time, all NP-complete problems can be reduced to any NP-hard problem in polynomial time. Then if there is a solution to one NP-hard problem in polynomial time, there is a solution to all NP problems in polynomial time.

The halting problem is the classic NP-hard problem. This is the problem that given a program P and input I, will it halt? This is a decision problem but it is not in NP. It is clear that any NP-complete problem can be reduced to this one.

P = NP : This the most famous problem in computer science, and one of the most important outstanding questions in the mathematical sciences. In fact, the Clay Institute is offering one million dollars for a solution to the problem (Stephen Cook's writeup on the Clay website is quite good). It's clear that P is a subset of NP. The open question is whether or not NP problems have deterministic polynomial time solutions. It is largely believed that they do not. Here is an outstanding recent article on the latest (and the importance) of the P = NP problem: The Status of the P versus NP problem.

The best book on the subject is Computers and Intractability by Garey and Johnson. My favorite NP-complete problem is the Minesweeper problem.