# P

P is a complexity class that represents the set of all decision problems 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 connected graph G, can its vertices be coloured using two colours so that no edge is monochromatic?

Algorithm: start with an arbitrary vertex, color it red and all of its neighbours blue and continue. Stop when you run out of vertices or you are forced to make an edge have both of its endpoints be the same color.

# NP

NP is a complexity class that represents the set of all decision problems for which the instances where 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 factorisation is in 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

NP-Complete is a complexity class which represents the set of all problems *X* in NP 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 y of Y to instances x = f(y) of X in polynomial time, with the property that the answer to y is yes, if and only if the answer to f(y) is yes.

**Example**

3-SAT. This is the problem wherein we are given a conjunction (ANDs) of 3-clause disjunctions (ORs), 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). 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 at least as hard as the NP-complete problems. Note that NP-hard problems do not have to be in NP, and 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.

**Example**

The halting problem is an 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. As another example, any NP-complete problem is NP-hard.

My favorite NP-complete problem is the Minesweeper problem.

# P = NP

This one is 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 write up 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.