J P, NP, and NP-completeness

J1 Problems in P

We consider “decision problems” for graphs. The input is a graph and maybe some other parameters (such as an integer $k$), collectively known as an “instance”, and there is a single yes/no question. Like: “Does the graph have an Eulerian circuit?” or “Does the graph have a clique of size $k$?”

An algorithm is a set of clear instructions for solving a problem: the instructions break down into a sequence of steps. For an algorithm for a graph problem, let $f(n)$ denote the maximum number of steps that the algorithm would take when confronted with a graph on $n$ vertices. The function $f(n)$ is known as the running time of the algorithm. Such an algorithm is said to be polynomial-time if there exists a polynomial $g(n)$ such that $f(n) \leq g(n)$ for all $n$. The set of all problems that can be solved in polynomial time is denoted by $P$. Colloquially people say such problems have “efficient” algorithms.

For example, consider the problem to determine if an input connected graph has an Eulerian circuit or not. Then a reasonable algorithm would simply calculate for each vertex the degree of that vertex and check that all degrees are even. Assume we are given the graph as an adjacency matrix. (The exact representation does not matter when we define $P$.) Then checking the parity of the degree of a particular vertex takes at most $n$ steps. And the whole algorithm takes about $n^2$ steps. Thus there is an efficient algorithm to test whether a connected graph has an Eulerian circuit or not.

Common graph decision problems in $P$ include:

- Is it bipartite?
- Is there a perfect matching?
- Is the distance between two specified vertices at most specified value?
- Is the graph $k$-connected?

There are of course many graph problems where the answer is a value rather than just yes–no.
J2 Problems in NP and NP-completeness

There are many graph problems for which efficient algorithms are not known. Suppose I know that the graph has a Hamilton cycle and I wish to convince you that there is one. Well I can simply tell you the Hamilton cycle and then you can easily verify that it is valid. But how would I prove to you that there was not a Hamilton cycle?

An instance of the problem for which the answer is yes is known as a yes-instance. For the problem of

**INPUT:** a graph.  **QUESTION:** Does the graph have a Hamilton cycle?

there are short *certificates* or “proofs” for yes-instances.

A problem is in NP if there exists an efficient algorithm which, given an instance $I$ and a potential certificate $C$, can tell whether the $C$ gives a valid proof that the answer is yes. It is important to note that: If the instance is a yes-instance, then there must exist a certificate that will convince the algorithm. If the instance is a no-instance, then there must be no certificate that will convince the algorithm.

Informally we say that a problem is in NP if there exists an efficient algorithm which with some prompting can verify a yes-answer. Obviously $P \subseteq NP$.

A major open question in mathematics today is whether $P = NP$ or not. Most researchers strongly suspect that there are problems in NP which are not in P. But we cannot prove it.

A problem is said to be **NP-complete** if the existence of an efficient algorithm for it would imply the existence of an efficient algorithm for every problem in NP. Essentially the NP-complete problems are the hardest ones in NP.

There are thousands of examples of NP-complete graph problems. These include:

- Does the graph have a Hamilton cycle?
- Is it 3-colorable?
- Is it decomposable into $K_3$’s?
- Is the independence number at least $k$?