$P = NP$?

Intro

Polynomial Time Computable Algorithms

Algorithms whose running time is bounded on every input of size n by cn^d (where c and d are constants).

Polynomial time algorithms are reasonably efficient relative to exponential time algorithms.

The P–NP distinction is between polynomial-time and exponential-time algorithms.

Polynomial Time Reductions

(Note: this work is from a book "Polynomial Time Reductions", section 8.1 – detailed reference in his notes, apparently copies are available in the library.)

We need some concept of things being "equivalent" to one another. Given two problems X and Y (e.g. sorting problems), we want to say that they are equally hard to compute. We'll express that if (and only if) one can be solved in polynomial time, so can the other.

We consider a black-box solution for a problem X, where we assume that the solution runs in polynomial time.

We write $Y \leq_P X$ to say that X is at least as hard to solve as Y – meaning that Y can be solved using X, but not necessarily that X can be solved using Y.

Given a black box solution to X, we're saying an arbitrary instance of Y (e.g. any particular list to sort) can be solved with a polynomial number of standard computation steps, plus a polynomial number of calls to X.

If (and only if) $Y \leq_P X$ and $X \leq_P Y$, then $Y =_P X$ – the two problems are polynomially equivalent.

Two Problems

Consider two problems:

1. Max independent subset of a graph.
2. Find the smallest vertex cover of a graph.

We looked at finding a max independent subset of a graph before.

A vertex cover is a subset (A) of the vertices (V) in a graph, such that the vertices in A cover the edges of the graph. I.e. every edge in the graph touches a vertex in the subset A.

These problems are polynomial-time equivalent. You can see the relation by noticing (or proving) that if you have a smallest vertex cover, all the vertices not included in the cover form a max independent subset.

$P = NP$?

P is the set of polynomial-time computable algorithms, and NP is the set of problems with polynomial-time checkable solutions.

The independent set problem $\in NP$.

Formal Definition of NP

An algorithm `A` for a decision problem `X` receives an input string `S` and returns `1` in the case that $s \in X$ and `0` otherwise.

`A` is polynomial-time if it terminates on each input `S` in $O(P(|S|))$, where `P` is a polynomial in `|S|`.

`P` is the set of all problems `X` for which there is a polynomial time algorithm `A` that solves `X`.

Efficient Certifiers

B is called an efficient certifier for problem X if the following holds:

1. B is a polynomial-time algorithm and takes two inputs: S and t.
2. There is a polynomial-time function P such that for any input string S:

   $S \in X \Leftrightarrow \exists t$ such that $|t| \leq P(|S|)$ and $B(S, t) =$ "yes"

Definition

NP is the set of all problems for which there exists an efficient certifier.

$$P \subset NP$$

Consider: Problem X in P.

X has a polynomial-time algorithm A that solves X.

Design B as follows:

[...]

NP-Completeness

An NP-complete problem X satisfies:

1. $X \in NP$
2. $\forall Y \in NP : Y \leq_P X$