Introduction

Algorithmic time complexity has interesting applications to Number Theory and much more. How long does it take to sort a list of a million numbers?

Can large primes be factored quickly? If the solution to a problem can be verified in polynomial time, can it be solved in polynomial time? [**?**].

If P=NP, then cryptographic security systems, as they stand now, are broken. It is as easy to decode encrypted messages as it to encode them.

Solving this problem is interesting and valuable because of its implications to data security as well as speeding up computers.

History

The P vs NP question is famous for being one of the most accessible millennium prize questions[**?**]. It began in 1936 when Alan Turing developed a theoretical computational model, which later became a useful theoretical model for computation. The theoretical computational model developed by Turing is sometimes referred to as a theory of computability. Basically, how long does it take to solve certain problems and are there ways to improve efficiency in solving a given problem. Although, the original model soon failed as the needs for time and memory were not accounted for[**?**].

Throughout the 1960's, 70's, and 80's the understanding of time complexity developed rapidly. Several mathematicians and computer scientists proved many problems could be solves in polynomial time and created notations that illustrated the time (and sometimes, space) complexity of their calculations. In the 1970's **NP**-completeness became a very popular subject in mathematics, because computers were on the rise and the search of algorithms to solve even the most basic problems, in the most efficient ways[**?**]. Remember that the

computers used to land a man on the moon had less power than phones had over a decade ago!

Notation

Some good rules for mathematical notation:

- P = Set of all problems which are solvable in polynomial time.
- NP = Set of all problems whose solutions can be verified in polynomial time.
- NP-Hard = Set of all problems that are at least as hard as the problems in NP.
- NP-Complete = Set of all problems that are in both NP and NP-Hard.