<http://news.mit.edu/2009/explainer-pnp>

Title of the Article: Explained: P VS NP

Name of Journal, Date of Publication: Explained: P VS NP, October 29, 2009

Primary Author(s): Larry Hardesty

Summary of the background:

The question of whether P equals NP remains one of the unsolved mysteries in computer science. It is also the most important outstanding question in computer science. Roughly, P is a set of relatively easy problems where NP is a set of hard problems.

Summary of the objectives:

The question of whether P equals or does not equal NP is asking if there are easy solutions to the hard problems. When people ask how long it takes to run the algorithm or the code, they usually give answer in minutes or seconds. However, in computer science, the answer is dependent on the number of elements the algorithm has to manipulate.

For example, if we an unsorted list of numbers and are asked to find the largest number using an algorithm, there might be complicated algorithms that take longer time than other algorithms since one can be more efficient compared to the other one. The algorithm’s execution time is proportional the number of elements, which we call it N. Some algorithms can be proportional to N while some can be proportional to N^2, or etc. The mathematical expression that involves N or N to the some power is called polynomial. The question of whether P equals or does not equal NP, in detail, is asking whether a set of problems is proportional to polynomials involving N. The algorithm which includes N to the power of a number is much slower compared to N itself. For example, it is obvious that N^3 takes more time to complete compared to N. However, there are worse scenarios where it is exponential, such as 2^N, which takes much longer time compared to any of N to the power of a number.

So, NP stands for nondeterministic polynomial time, where they are problems whose answers can be answered within polynomial time, but take too long due to the exponential polynomial time.

Summary of the discussion:

To be more clear, the question of Does P equal NP is asking whether the “solution to a problem can be verified in polynomial time, can it be found in polynomial time?” NP problems that take exponential amount of time are called NP-complete problems, which include fairly common scheduling tasks. One of the most famous questions is the traveling-salesman problem: given N cities and distances between those cities, is there a route that hits all of them but is shorter and is the most efficient for salesman?, which seems easy but has been unanswered in polynomial time. Solving the NP vs P problem will be very significant in the field of computer science. According to Sipser, a member of Computer Science and Artificial Intelligence Lab’s Theory of Computation Group, states that some algorithms for NP-complete problems are in exponential level only in the worst-case scenarios and that in average cases, they are more efficient than polynomial-time algorithms.

What did you learn from article? What is most pertinent for your research?

From this article, I learned the importance of algorithm since the algorithm is the one in computer science that chooses the speed of the program, depending on whether it is N, N^2, or exponential.

What questions do you have about the content that remain unanswered?

Some questions that I have still have after reading this article are : There are many NP problems that have not been solved in polynomial time. In order to prove P vs NP, do all of the problems need to be proven to be answered in polynomial time or not? How do we prove it?