Rayaan J. Irani and Saaketh Koka

Professor David Brauchler III

CS 2353

22 Apr. 2021

Does P = NP?

Since the early 1970s many Computer Scientists all around the world have dedicated many years and resources to answer the question: Does P = NP? If someone were to find the answer to this question, many fundamental aspects of Computer Science and programming would be changed forever.

NP complete problems (nondeterministic polynomial time problems) are very hard to compute but very easy to verify if the solution to such a problem was correct. P problems (polynomial time problems), however, can be both easily solved and easily verified. The question, does P = NP, can be re stated as: if a solution to a problem is easily verifiable (is verified in polynomial time), is there always an easy way to solve this problem (solvable in polynomial time)?

So, why does this question matter? Well, it is simple. If P = NP, then there would exist a way to simplify all equations that run in nondeterministic polynomial time to polynomial time thus opening up a multitude of different problem-solving techniques and revolutionizing computing as we know it. While many Scientists debate on whether or not the if someone was to prove that P=NP, there would be drastic consequences, both positive and negative. For example, if P did equal NP, and an algorithm was created to solve protein folding problems in polynomial time, then scientists can easily study and find cures for many diseases such as cancer or even Covid-19 in a short period of time. On the other hand, if P was equal to NP, then cracking passwords would become very easy, hackers would be able to very easily bypass common security measures and likely cause lots of harm to the globally connected world.

Now coming back to the original question, does P = NP? Well as of today there is no answer. It has yet to be proved or disproved. In a survey conducted by Lane A. Hemaspaandra, only 5 out of the 79 respondents, with a background in computing, believed that the question would never be resolved and forever remain a mystery. Further, 36 of the 79 respondents believe that the technique used to resolve the question “is known to us now, but not the way to apply it” (Hemaspaandra 3). From these responses it is clear that many experts believe that this question will be resolved in the next few decades.

In order to prove that P=NP, you would need to prove that all problems that take non-deterministic polynomial time can be configured to run in polynomial time. Many times throughout history a few problems that were originally thought to run in non-deterministic polynomial time turned out to be able to run in polynomial time by rethinking the algorithm used to solve the problem. This, however, does not prove that P = NP. In order to prove that all problems in the set of NP are in the set of P, we need to find a way to prove that *every* item in the set of NP is in P not just a couple here and there. On the other hand, in order to prove that not all elements in the set of NP are in P, we just need to prove that a single element in the set of NP is not in the set of P. With that being said, proving either of these cases will be very difficult.

This problem is further confounded by the fact that providing a proof to a problem is itself an NP complete problem as it takes a non-deterministic problem to generate a proof but, it is a relatively easy feat to verify a proof is valid. When reflecting on this fact, Scott Aaronson noted, “If P=NP … everyone who could enjoy a symphony would be Mozart” (“P vs. NP” 10:08). We believe that everyone can be Mozart. However, we leave that proof to the reader.

Work Cited

Hemaspaandra. SIGACT News Complexity Theory Column 36. University of Rochester. doi:

10.1145/564585.564599.

hackerdashery. “P Vs. NP and the Computational Complexity Zoo.” YouTube, YouTube, 26

Aug. 2014, www.youtube.com/watch?v=YX40hbAHx3s.