A Personal View of the P versus NP Problem

Lance Fortnow

Georgia Institute of Techology, Atlanta, Georgia 30309 USA,
fortnow@cc.gatech.edu,
http://lance.fortnow.com

I recently completed a general audience book on the P versus NP problem [1]. Writing the book has forced me to step back and take a fresh look at the question from a non-technical point of view. There are really two different P versus NP problems. One is the formal mathematical question, first formulated by Steve Cook in 1971 [2] and listed as one of the six unresolved millennium problems by the Clay Mathematics Institute[1]. The other P versus NP problem is the one that interests physicists, biologists, economists and the mathematically-curious general public. This talk will explore both faces of the P versus NP problem and what it means for mathematics and computer science moving forward.

Most of this audience is familiar with P versus NP as a mathematical challenge. Let Σ^* be the set of finite length sequences ("strings") over an alphabet Σ. Typically we take $\Sigma = \{0, 1\}$. Let M be a computational procedure that takes as input a finite string and either "accepts" or "rejects" the string. Typically we describe M by a Turing machine, a computational device developed by Alan Turing in his classic paper [3]. A machine M computes a language $L \subseteq \Sigma^*$ if for all x in L, M on input x accepts and for all x not in L, M on input x does not accept. The machine M runs in polynomial time if there is a fixed polynomial p such that for all inputs x, M on input x accepts in time at most $p(|x|)$ where $|x|$ is the number of alphabet symbols in the string x.

By default M is deterministic, the next action of the machine is uniquely defined by its current configuration. We can also consider nondeterministic Turing machines M that have many possible legal future actions. We say a nondeterministic machine M accepts if there exists a series of legal actions leading to M starting on input x to an accept state.

The class P is the set of languages L that are computable in polynomial time by some (deterministic) Turing machine. The class NP consists of those languages accepted by nondeterministic Turing machines in polynomial time. The P versus NP questions asks simply if P = NP.

The question immediately became central to theoretical computer science for several reasons. First of all, the question is quite robust to the various models. It doesn't matter which model of Turing machine we use, or we can use any other reasonable model of computation, the answer to the P versus NP question will not change. More importantly Cook [2], Karp [4] and Levin [5] identified a number of logical and combinatorial problems that are NP-complete, computationally as hard as any problem in NP, and thus have polynomial-time algorithms if and only if P = NP.

[1] http://www.claymath.org/millennium

This talk will explore some of the approaches researchers have used to attack the P versus NP problem, though we don't hold out much hope for a quick resolution.

We will also explore the other less mathematical side of P versus NP question. Computational power has dramatically increased through the years since Cook and Levin first formulated the P versus NP problem in 1971 allow us to solve via clever algorithms and brute force search a number of moderate-sized NP-complete problems. Paradoxically, these successes have only bolstered interest in the P versus NP problem, as we now wish to tackle larger computational problems where the exponential running times of our best algorithms for NP-complete problems really kick in. The huge growth in data has also made the P versus NP problem an issue for scientists in many fields such as biology, physics and economics.

These scientists and the general public don't care so much about the fine technical details of the P versus NP problem but more about the spirit of what we can or cannot solve on computers in a reasonable amount of time. In this talk we look at a "beautiful world" (with hidden dangers) that arise if P = NP (in a strong way).

While most computer scientists believe P $\neq$ NP, that can't be the end of the story as we need to tackle these difficult problems. The talk will give a high-level survey of some approaches to dealing with NP-complete problems such as heuristical and approximate algorithms.

The P versus NP problem is both an incredibly challenging mathematical puzzle but simply understanding the P versus NP problem and it challenges is now of critical importance for any scientist. Our challenge is to tackle the mathematical challenge while keeping the practical issues at heart.

References

1. Fortnow, L.: The Golden Ticket: P, NP and the search for the impossible. Princeton University Press, Princeton (2013)
2. Cook, S.: The complexity of theorem-proving procedures. In: Proceedings of the 3rd ACM Symposium on the Theory of Computing. ACM, New York (1971) 151–158
3. Turing, A.: On computable numbers, with an application to the Etscheidungs problem. Proceedings of the London Mathematical Society **42** (1936) 230–265
4. Karp, R.: Reducibility among combinatorial problems. In Miller, R., Thatcher, J., eds.: Complexity of Computer Computations. Plenum Press (1972) 85–103
5. Levin, L.: Universal'nyĭe perebornyĭe zadachi (Universal search problems: in Russian). Problemy Peredachi Informatsii **9**(3) (1973) 265–266 Corrected English translation in [6].
6. Trakhtenbrot, R.: A survey of Russian approaches to *Perebor* (brute-force search) algorithms. Annals of the History of Computing **6**(4) (1984) 384–400