NP != NOT-P

When I first started learning about algorithmic complexity and intractability, I remember struggling with the concept of P vs. NP (see **here** for a primer). Part of this, I now realize, was due to the confusing acronyms that have been assigned to the problem: when I see P and learn that it refers to “problems solvable in polynomial time”, I immediately assume that NP must refer to **non**-“problems solvable in polynomial time,” i.e., problems for which a solution cannot be found in polynomial time.

The thing is, that’s not actually the definition of NP: the N does not stand for **not**, it stands for **non-deterministic**. NP is not **not**-P; it’s **non-deterministic**-P.

To be more specific: P is the set of problems whose solution can be **found** in polynomial time; NP is the set of problems whose solution can be **verified** in polynomial time.

SAT

Lets look at these definitions in action. Take SAT, which is defined as, “given a statement in conjunctive normal form, can you find a satisfying assignment?” For example, if I give you the statement (A || B) && (!C), one solution would be (A = true), (B = true), (C = false), because this assignment satisfies the statement.

Now, no polynomial-time algorithm is known for finding a solution to a SAT problem. Thus, as of now, it’s unsure whether or not SAT is in P (because we don’t know how to **find** a problem in polynomial time). However, SAT is definitely in NP (because we know how to **verify** a solution in polynomial time)—I can give you an assignment, and you can plug it into the statement to see if it’s satisfied.

Hopefully that makes things a little clearer.

P = NP?

The P = NP? Problem is one of the Clay Mathematical Institute’s Millenium Prizes: if you solve it, you get $1,000,000 (so it must be important). The question boils down to this: are all problesm in NP also in P? I.e., for all problems whose solutions we can **verify** in polynomial time, does there exist a way to **find** a solution in polynomial time?

What a lot of CS newcomers don’t see immediately is that P is *definitely* a subset of NP: any problem for which we can find a solution in polynomial time can definitely have its solution verified in polynomial time. Regardless of whether or not you think P = NP, this is true. The question is whether the P is a perfect subset of NP: are all problems in NP also in P?

If you think that P != NP, then a visual representation of your belief looks something like this:

Conversely, for P = NP:

Significance

This is (almost indisputably) the most important open problem in computer science. If P = NP, then there would be countless problems which were once intractable, yet would now be deemed easily solvable—suddenly, all of these very, very, very difficult problems in NP that were seen as exponentially hard would be solvable in polynomial time.

It’s kind of hard to believe that this could be possible. I think **Scott Aaronson** puts it best:

“If P = NP, then the world would be a profoundly different place than we usually assume it to be. There would be no special value in "creative leaps," no fundamental gap between solving a problem and recognizing the solution once it's found. Everyone who could appreciate a symphony would be Mozart; everyone who could follow a step-by-step argument would be Gauss.”

In essence, he’s saying that P = NP would imply that solving a problem is as easy as verifying a solution.

**Finding a solution to a Sodoku board would be as easy as checking that a solution is correct—this just doesn’t sound right, does it?**

Right now, the industry leans heavily towards P != NP, but the problem is still open and no one has come up with a sufficient proof to claim the Millenium Prize. Personally, I tend to agree with Aaronson and the link—but I’d love to be shocked.