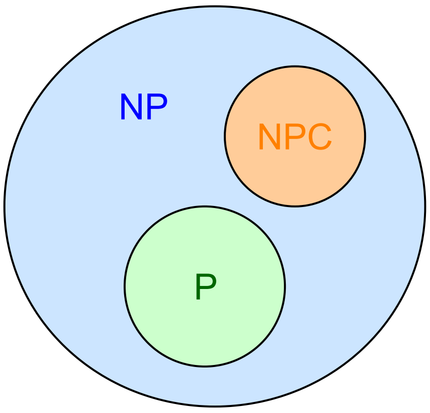
Between P and NP ?

Do any problems exist that aren't NP-complete and not in P?



Assume P is not equal to NP

NP-Intermediate

Given the above diagrams and assuming P≠NP, one may wonder if there are any problems that are in NP but not in NPC or P? Note that we have assumed **P≠NP**. Problems that exist in this class are called ***NP-intermediate***. Again, very carefully note that NP-intermediate exists only if P≠NP! For the interested, *Ladner's theorem* proves that if P≠NP then NP-intermediate is non-empty.

Prime Factorization

Prime factorization is ***suspected*** to be NP-intermediate problem. Note the use of the word suspected! No one has been able to come up with a polynomial time algorithm to find prime factors for a given integer, but that doesn't mean none exist! The converse hasn't been proven either, that no polynomial time algorithm exists for prime factorization of integers. Moreover, no one has been able to reduce it to a NP-complete problem, and thus prime factorization can't be placed in the NP-complete complexity class. As a result, integer factorization is thought to likely be an NP-intermediate problem.

Beyond NP-Hard

There are several other complexity classes based on different criterion; e.g. PSPACE, 2-EXPTIME, PCP and a bunch more are listed here. However, as

software engineers in the industry, knowledge of P, NP and NP-complete problems is sufficient for most purposes. It helps to recognize when a problem may not yield to any polynomial algorithm design technique. Likewise, even informally proving the problem to be NP-complete helps direct energies towards an approximation algorithm or a heuristic rather than mindlessly banging one's head against the wall for a polynomial solution. Some of the problems that one can see as professional software engineers and are NP-complete include

- Scheduling problems, e.g. doctors' duty roster at a hospital. These problems usually involve combinatorial complexity.
- Google Maps generating best route for multiple destinations. Read more here.
- Development of compilers and languages involves questions like if a grammar can generate an ambiguous language, which can't be solved.

Advancements are always afoot in the research community to better our understanding of computational complexity of algorithms. As a recent example, determining if a given integer is prime was thought to be an NP-intermediate decision problem until 2012. At that time, three Indian researchers were able to show that there did indeed exist a polynomial time algorithm for primality test, and the problem was moved to the P complexity class.