Class P and NP problems. Complexity class diagram of decision problems.

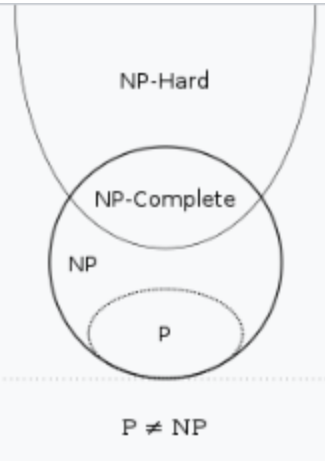
P vs NP

- The class P consists of those problems that are solvable in polynomial time.
- More specifically, they are problems that can be solved in time $O(n^k)$ for some constant k, where n is the size of the input to the problem.
- The class NP consists of those problems that are “verifiable” in polynomial time.
- “verifiable” means that if we are given a solution to a problem then we are able to verify that this proposed solution is indeed correct in polynomial time.
- In general, when we consider P, NP, NPC, etc. problems we are only focusing on decision problems, however as we know from previous notes we can reduce many optimization problems into decision problems, which means it also applies to them.
- Any problem in P is also in NP, since if a problem is in P then we can solve it in polynomial time without even guessing the solution.
- The open question is whether or not P is a actually equal to NP, so far we are only able to tell that $P \subseteq NP$.

NP-complete (NPC) and NP-hard

- NP-Complete is a complexity class which represents the set of all problems `X` in NP for which it is possible to reduce any other NP problem `Y` to `X` in polynomial time.
- This means that we can solve any NP problem in polynomial time if we know how to solve any NPC problem in polynomial time.

- Remark: There is another complexity class called NP-hard which represents all problems for which there exists a polynomial transformation from any problem in NP into the problem from this class. The NPC is an intersection of NP-hard and NP, since an NP-hard problem by definition doesn't need to be in NP, NPC problems are those problems that are both NP-hard and NP.



This illustration assumes P is not equal to NP

- NPC problems are very important since if any NPC problem were solved in polynomial time, then every problem in NP would also be solved in polynomial time, which would proof P=NP.

Weakly NPC and Strongly NPC

- A problem in said to be weakly NPC if it's NPC and there exists a pseudo-polynomial time algorithm that solves it.
- A problem is said to be strongly NPC (SNPC) if it's NPC and there doesn't exists a pseudo-polynomial time algorithm that solves it.
- In other words, if a problem can only be solved in polynomial time using a non-deterministic machine, but there exist an algorithm that can solve it in polynomial time with respect to the input value not the input size on deterministic machine then it is said to be weakly NPC.
- A strongly NPC problem is a problem that remain NPC even if we use unary encoding.

- Examples of weakly NPC problems: Knapsack, Set partition, P2 || Cmax
- Examples of strongly NPC problems: Traveling salesman, Clique, Vertex cover, 3-dimensional matching, Hamiltonian circuit.