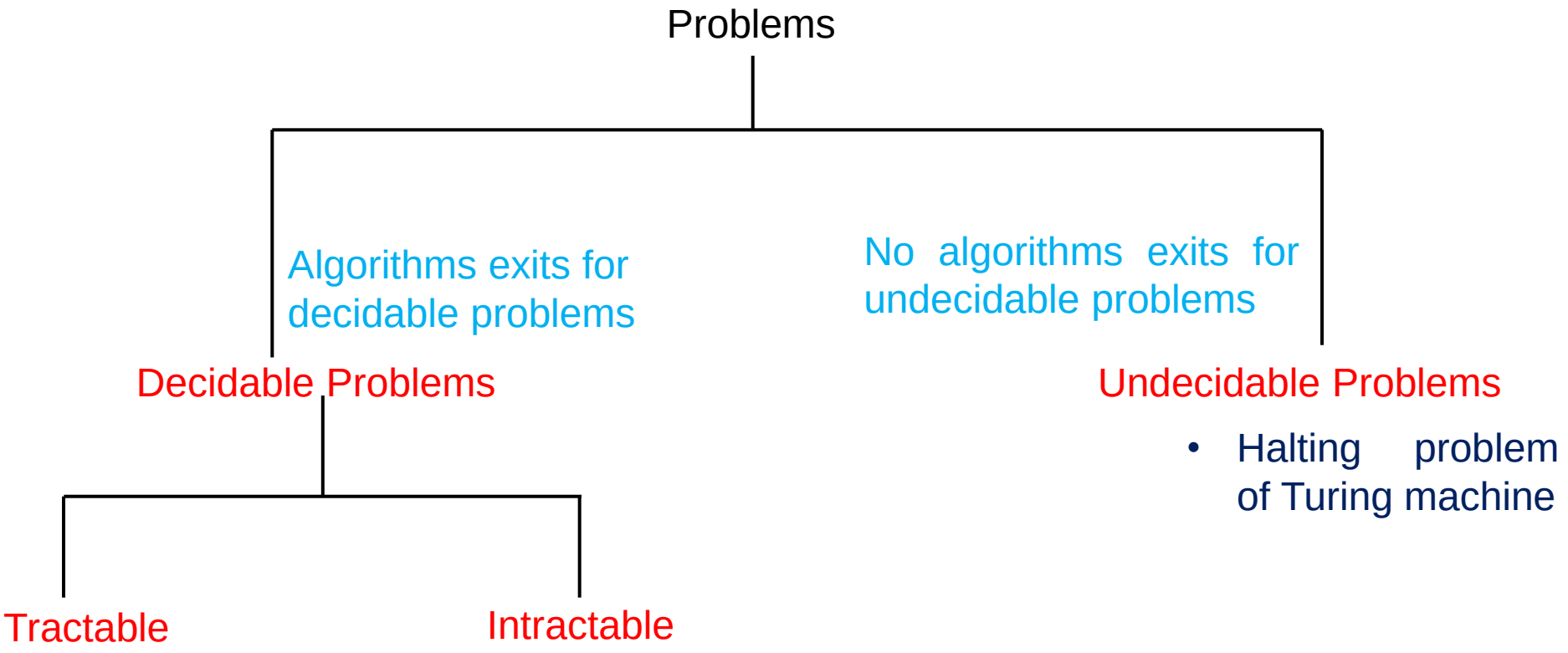
NP Problems

Types of Problems



Undecidable Problems are problems for which no algorithms exist, which means, there is no procedure/ algorithms for the problem that halt after certain period of time.
For example: Halting problem of Turing machine

Tractable Problems

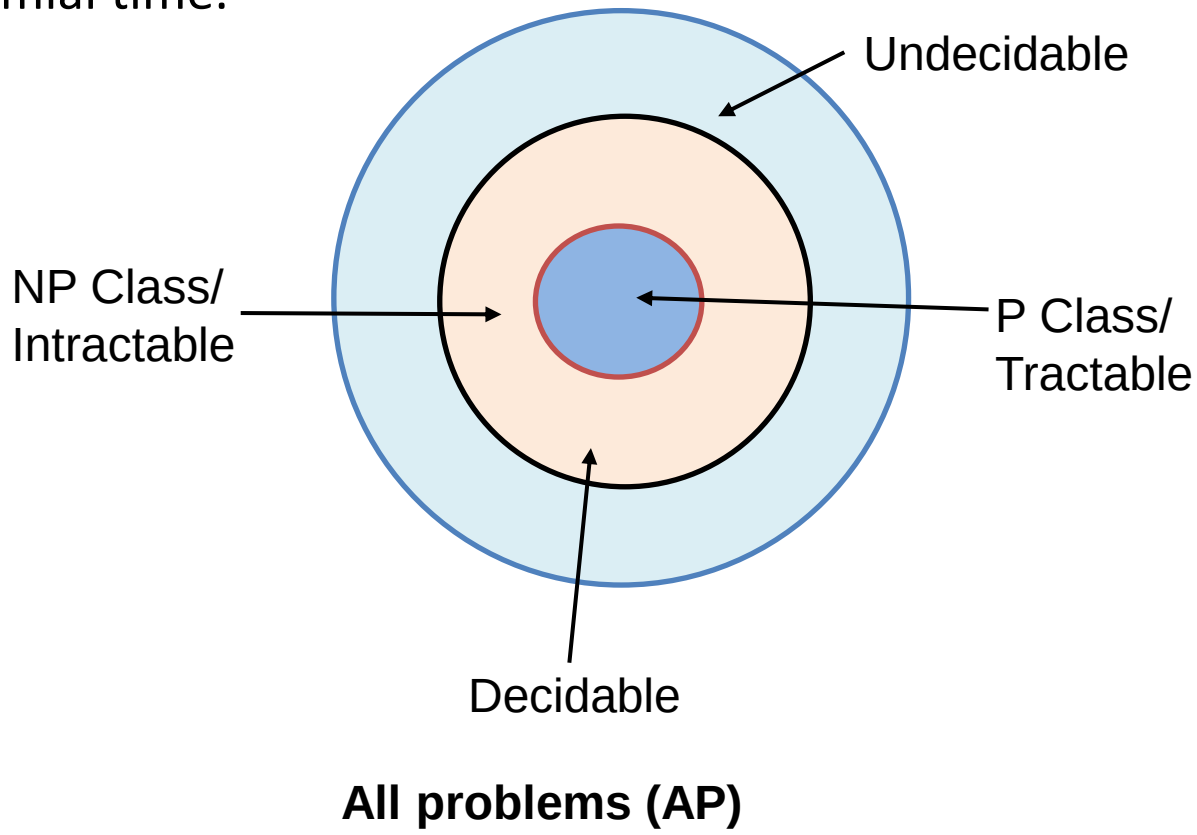
- **Tractable:** Problems that can be solved in a reasonable (polynomial) time, i.e., $O(n^k)$.
- In some cases, the value of k could be very large.
- Tractable problems are also known as P-class problems.
  - For example,
    - Linear Search
    - Binary Search
    - Merge sort
    - Quick sort
    - Job sequencing
    - Shortest path algorithms,
    - Etc.

Intractable Problems

- **Intractable:** Some problems are intractable, as they grow large, we are unable to solve them in polynomial time, i.e., $O(C^k)$.
- Intractable problems are also known as NP-class problems.
- The computational complexity increases with k value and sometime these problem keep running for months and don't produce any result.
- For these types of problems, many heuristic algorithms have been presented that produce approximate/ closed results in polynomial time.
  - For example,
    - Fibonacci series
    - 0/1 Knapsack
    - TSP
    - Etc.

Types of Problems

We cannot definitively claim that problems in the NP class are unsolvable in polynomial time. It remains a possibility that a future researcher could propose an efficient polynomial time solution for any NP problem. Therefore, we cannot conclusively assert that NP class problems are inherently unsolvable in polynomial time.



Optimization Problems

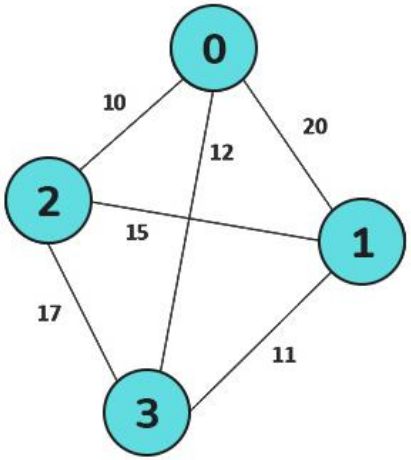
Problems which are decidable and don't belong to P-class are difficult to solve, since at present there is no polynomial time algorithms exist for them.

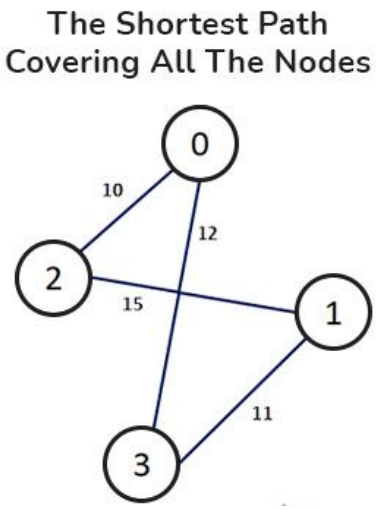
If anyone can prove that an intractable problem, can never be solved in polynomial time then only it can be concluded that they belong to NP class. However, finding such a prove is also a difficult task.

Let us consider some intractable optimization problems and prove that they cannot be solved in polynomial time.

- TSP
- 0/1 Knapsack

Optimization Problems





TSP can be solved in exponential time with the best-known algorithm i.e., dynamic programming.

Similarly, 0/1 knapsack also takes exponential time if the capacity of the knapsack is vey large.

At present, the abovementioned optimization problems cannot be solved in polynomial time. So, they belongs to intractable category and are difficult to solve, but how to show that the above problems are hard/ difficult to solve.

Optimization Problems

If we take a simpler version of TSP and 0/1 knapsack problem and prove that they are difficult to solve than it can be easily proven that the standard TSP and 0/1 knapsack is much harder/difficult to solve.

For TSP-

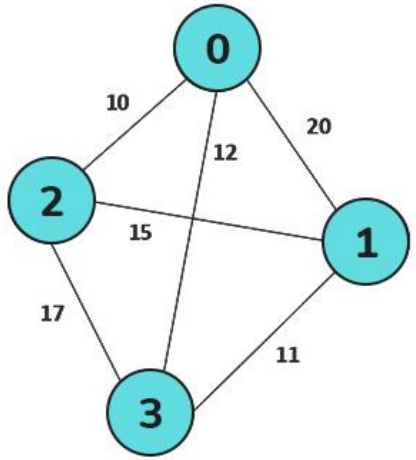
Is there any shortest path covering all vertices of length at most k?

The answer to the above question will be either yes or no.

Is there any shortest path covering all vertices of length at most 60?

If the simpler problem cannot be solved in polynomial time, then the standard TSP can never be solved in polynomial time.

The simpler problem instantiated from the original problem is known as **Decision Problems**.



Optimization Problems

Decision Problems - A decision problem is one with yes/no answer. These problems are normally used to verify the NP problems in polynomial time.

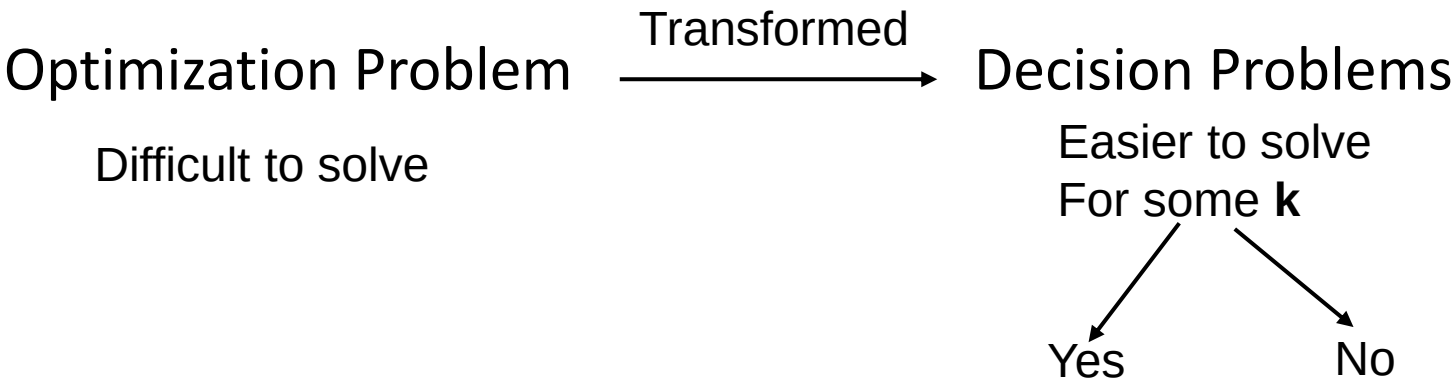
It is recommended to first solve decision problems before solving the intractable optimization problems.

Therefore, first a decision problem is formulated corresponding to each optimization problem.

For 0/1 knapsack the decision problem will be-

Is there any optimization problem whose profit is at least k?

Optimization and Decision Problems

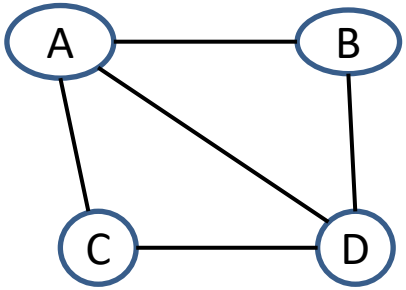


If the optimization problem is easy, then decision will also easy.

If we can prove that the **decision problem** is hard then **optimization problem** will certainly hard.

Decision problems need to verified before taking any decision. To verify these problems, verification algorithm is used.

Verification Algorithms



Is this graph Hamiltonian?

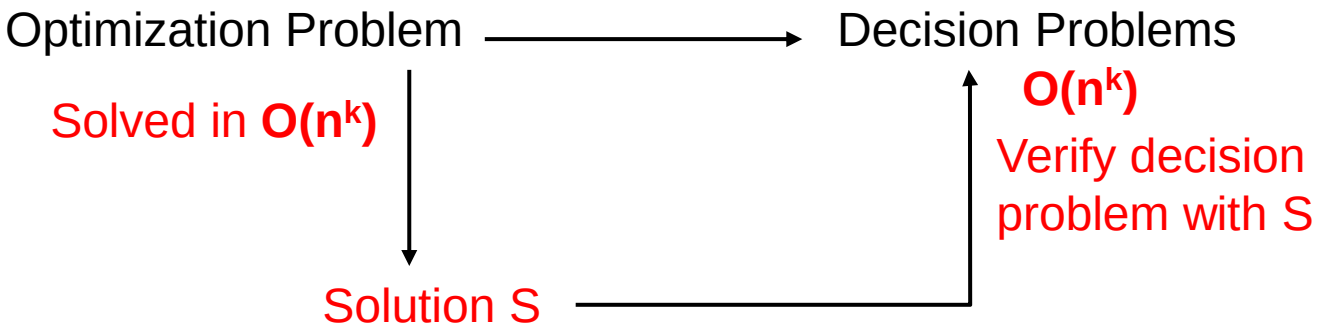
If a graph with decision problem and some solution is given.
Can we verify that the given solution is correct for decision problem in polynomial time?

For example, suppose A-B-D-C-A is given as a solution. In that case, the verification algorithm for the Hamiltonian cycle will check whether all vertices are present and whether there is an edge b/w two vertices in the original graph. Based on the steps of verification algorithm, answer of decision problem is identified.
Answer for the verification algorithm for this example is **YES**.

Verification Algorithms

As we know that intractable problems cannot be solved in polynomial time, but if someone claims that he/she has solved an intractable problem in polynomial time, then to verify whether their claim is right or wrong, verification algorithm is used. If the verification algorithm verifies their solution in polynomial time, then we can say that intractable problem may be solved in polynomial time else it cannot.

In simpler words, we can say that verification algorithm check whether given solution is right or wrong.



So, we can say, if the optimization problem is easy, then decision will also easy.

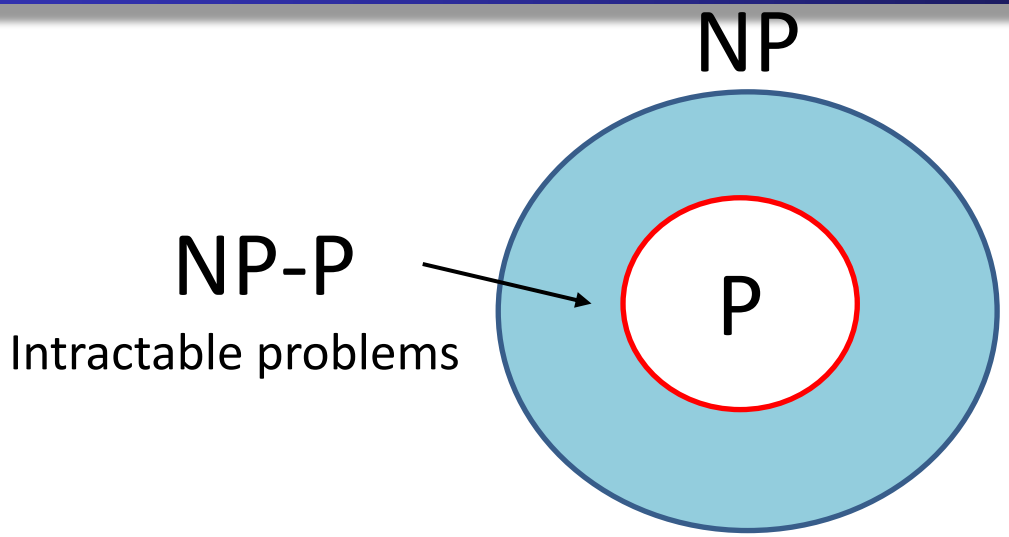
NP and P Class Problems

P- set of all decision problems for which polynomial time algorithm exist to solve them (quickly solved)

NP- set of all decision problems for which, there is a polynomial time verification (easily verified)

P Class Example	NP Class Example
MST of weight at most k for a Graph G Ans: Yes/ No (**Yes** using Prims and Kruskal algorithm)	MST and fractional knapsack problem can be solved in polynomial time, then obviously it can be verified in polynomial time. So, each problem which belongs to P will also be in NP + Some extra problem which will only belong to NP
Profit is at least k for a fractional knapsack of capacity C Ans: Yes/ No (**Yes** using Greedy algorithm)	
	TSP: no polynomial time algorithm to solve it but it can be verified in polynomial time, if some solution is given
	0/1 Knapsack

NP and P Class Problems



It has been observed from the previous example that each **P class problem** will also be an **NP class problem**. However, no information is available for **NP-P** class problems (intractable problems) that they will never be solved in polynomial time. If anyone in the future solves any of the **NP-P** problems, they can be moved to **P class** problems.

Many researchers believe that $P \subseteq NP$, but no proof exists. Similarly, it is hard to prove P=NP, (this question is still open).

NP and P Class Problems

1. It can be proved P=NP, if all problem outside of the P can be solved in polynomial time.

OR

2. If any (one or more) problem outside of the P cannot be solved in polynomial time, then we can say P!=NP, and it is proper subset of NP, i.e., $P \subseteq NP$.

However, the above statements still need to be proved.

The statement 1 is very hard/ impossible to prove since, there may be millions of problems in NP-P class and each one must be solved in polynomial time to prove P=NP.

To simplify it, we can find a problem from the NP-P class that is the toughest of all and try to solve it. If the problem is solvable in polynomial time, then P=NP can be proven. This entire process is called reducibility.

Polynomial Time Reduction

Any problem A is said to be polynomial time reducible to a problem B –

- If instance α of A can be transformed to some instance β of B in polynomial time.
- Answer to α is 'YES' if and only if answer to β is 'YES'

$$A \xrightarrow[O(n^k)]{\text{transformed}} B$$

The following information will hold for the above statements-

If A is **not in P**, then B is also **not in P**

OR

If A is hard B is hard

If A is **easy**, then B is also **easy**

OR

If A is **in P**, then B is also **in P**

Polynomial Time Reduction

A: Given n Boolean variables with values X1, X2, …, Xn. Does at least one variable have value true?

B: Given n integers i1, i2, i3, …, in. Is max(i1, i2, i3, …, in)>0?

Decision Problems

Consider n=4

For problem A

Suppose value are {T, F, F, T}

Does at least one variable have value true? YES

For problem B

Suppose value are {-30, 10, 0, 2}

Is max(i1, i2, i3, …, in)>0? YES

Let us talk about conversion

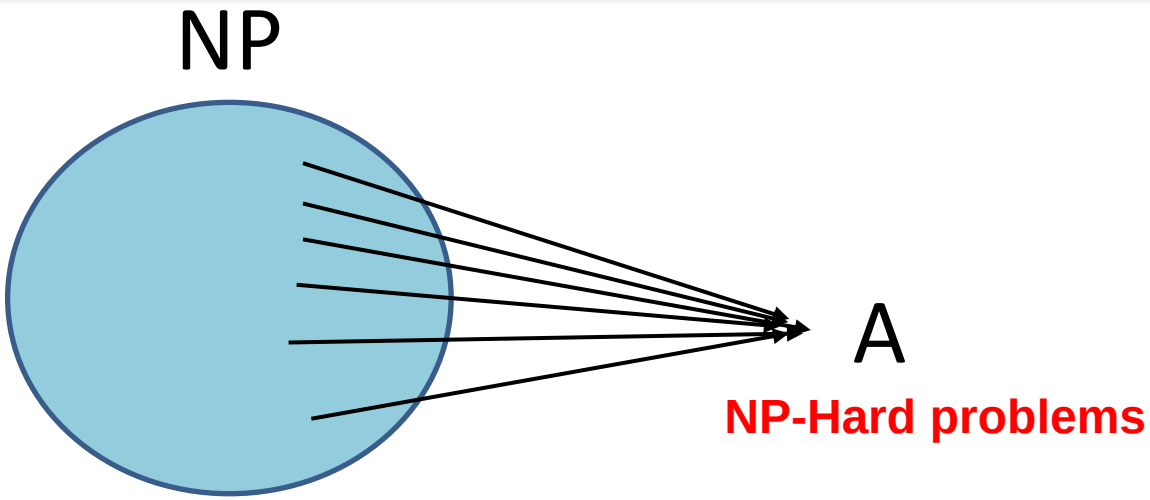
Can problem B be solved in polynomial time? YES **O(n)**

Scan value **{T, F, F, T}** of problem A and transform T to 1 and F to 0 to get an Instance of problem B.

{1, 0, 0, 1}

conversion done in **O(n)**

NP Hard

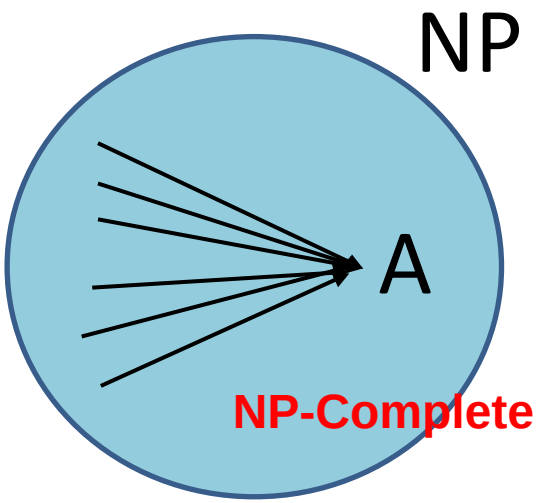


Suppose every problem which is in NP could be converted to another problem A in polynomial time and, if we could solve problem A in polynomial time then every problem in NP could also be solved in polynomial time.

These types of problems are known as NP-Hard problems, which means problem A is as hard as all other problems in NP. If this condition holds, then **P=NP.** However, this is not yet proven by anyone.

If the problem A cannot be solved in polynomial time, then it will be considered as NP-Hard.

NP Completeness

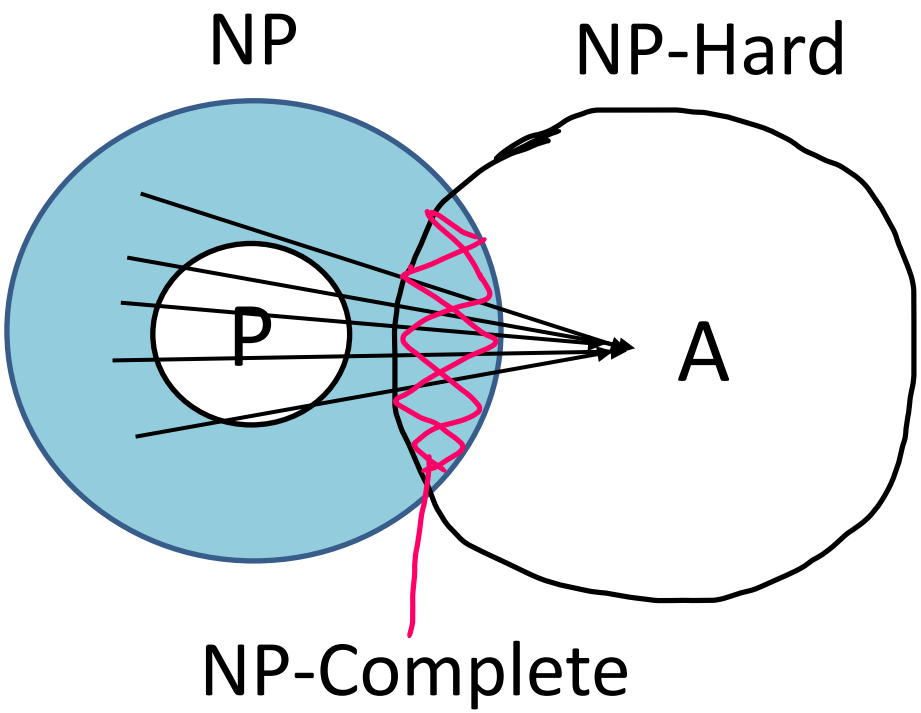


Suppose every problem which is in NP could be converted to another problem A which is also in NP (its solution can be verified in polynomial time), then problem A belongs to both NP and NP Hard (problem is as hard as other problems in NP). These types of problems are known as NP-Completeness.

NP, P, NP Hard and NP Completeness

After analyzing the literature, researchers started believing that there are some NP-class problems that can never be solved in polynomial time. However, they may be proven wrong if someone solves all NP problems in polynomial time.

The relationship among P, NP, NP-H, and NP-C is presented in the following diagram.



NP, P, NP Hard and NP Completeness

To prove NP=P, not every NP problem needs to be solved. If anyone can solve an NP-complete problem in polynomial time, then the claim NP=P can be proven.
If NP-hard or NP-complete problem is solved in polynomial time, then **NP=P.**

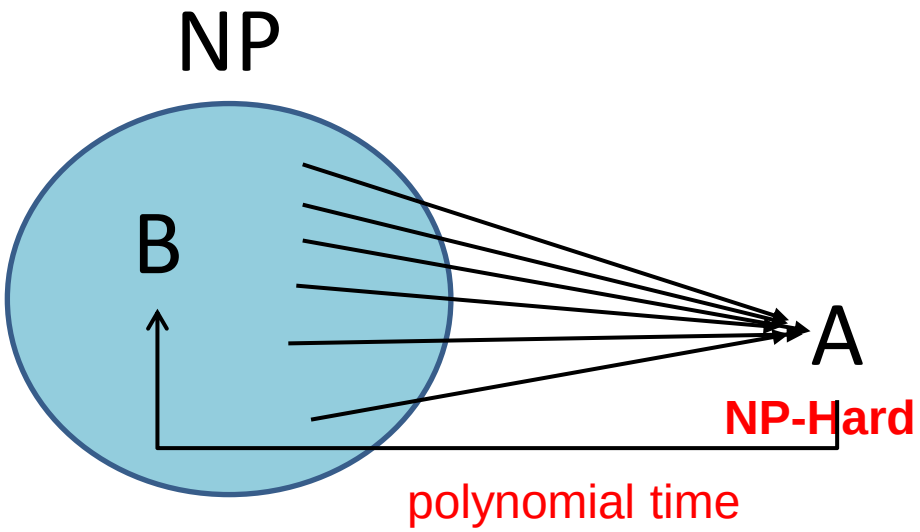
On the contrary, to prove **P!=NP**, take an **NP-complete** problem and show that it can never be solved in polynomial time.
If NP-problem or NP-complete problem is not solvable in polynomial time, then **P!=NP**.

Status of NP problems are still unknown.

NP, P, NP Hard and NP Completeness

If A is NP-hard and A is converted to B in polynomial time, then B is also NP-hard.

if A is NP-hard and B is NP and there is polynomial time conversion from A to B, then B is a NP-complete problem.



Well Known NP Problems