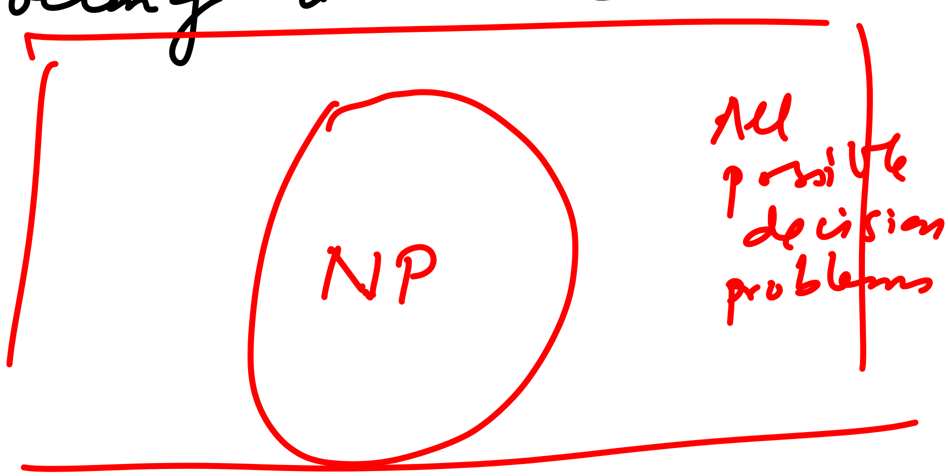
CSL 630 Lecture 27, Nov 13

NP algorithms -

- Make a guess (certificate) (non-deterministic step) : size of certificate is polynomial in n
- Verify in polynomial time

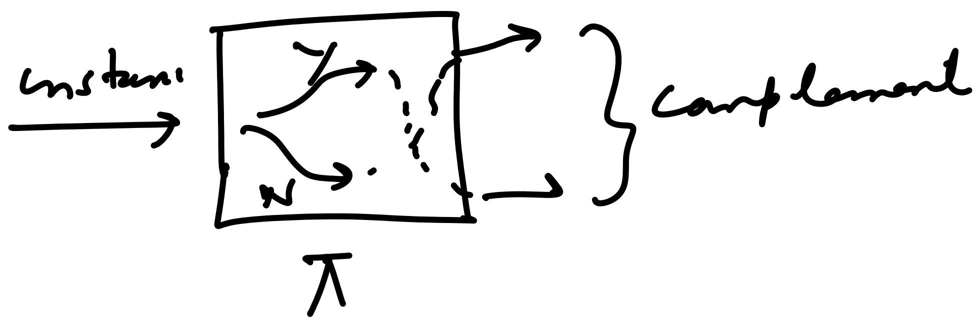
Any problem (decision problems) for which we can solve the "YES" instance using an NP algorithm belong to the class NP



However the complement of the problem, i.e. we flip the YES/NO may not have an NP algorithm

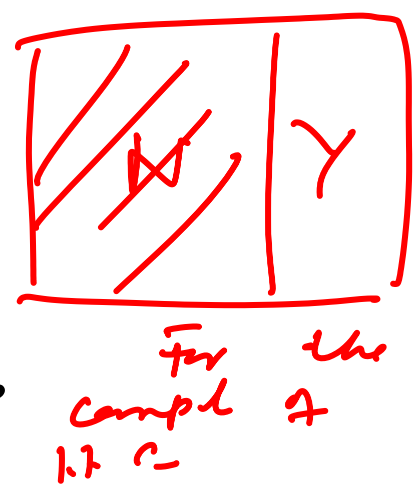
Note: Normal polynomial time algorithms belong to the class NP: (No guess is required)

This is the class P. For problems in class P, the complement is also in P

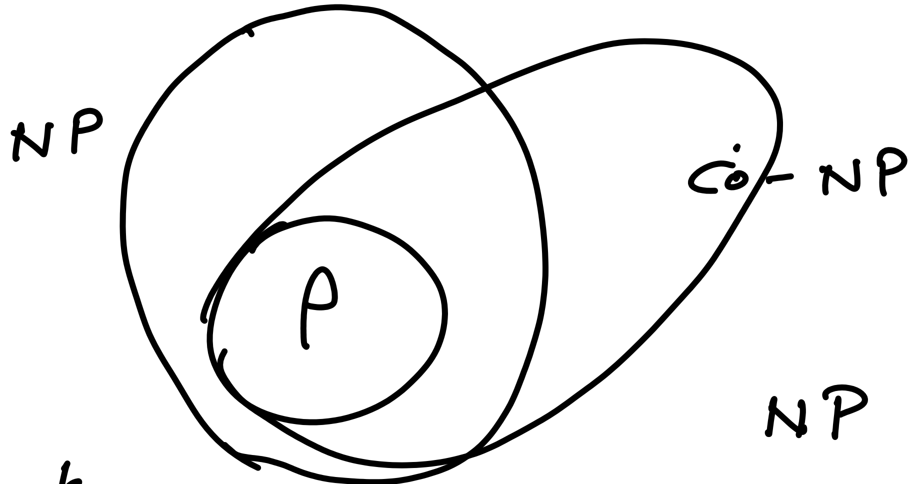


Decision Problem can be thought of as a set of strings (encodes the input instances) for which the answer is Yes

Example Hamilton cycle problem



The problems for which the complement is in NP are called co-NP class.



We know $P \subset NP$

$NP \stackrel{?}{=} P$

If $P = NP$

$\Rightarrow NP = \text{co-}NP$

Either $P = NP$, $P \neq NP$

Consider the "hardest" problems in NP and design a polynomial-time algorithm for it

NP complete problems

Show some non-polynomial lower bound for some problem in NP

NP complete problems (NPC)

- The problem is in NP
  (given & verify algorithm in polynomial time)

- All problems in NP can be reduced to this problem in polynomial time.

Sorting vs Selection

Selection is reducible to sorting

Given an instance of a selection problem
(a set of elements and a rank $k<n$)

we can create an instance of the sorting (a set of elements)

$$I_1 \xrightarrow{f} I_2$$

So that by solving the sorting problem I_2, we can find the soln for I_1

The reduction function (mapping) f should be efficiently computable (polynomial time), so that solving I_1 using the algorithm of I_2 is an efficient procedure

The reduction relation is denoted by $I_1 \propto_{t(n)} I_2$ ← time for reduction

This reduction may not be symmetric

$I_2 \overset{?}{\propto} I_1$

Can we reduce sorting to selection?

Note: The definition of "reduction" does not allow us to call the algorithm for I_2 more than once (many-to-reduction)

- when we allowed repeated calls
Turing reduction

Polynomial time reduction:

I/ The mapping function is computable in polynomial-time, then we say that problem π_1 is polynomial-time reducible to π_2

Defn : If all problems $\pi \in NP$ are polynomial-time reducible to some problem $\pi' \in NP$ then π' is called NP complete.

Claim : If π' is NPC and π' can be solved in polynomial time then $P = NP$

Do we have NPC problems?

How do establish some problem in NPC?

Cook-Levin theorem

The boolean satisfiability problem is NP complete

Given a boolean formula

$$x_1 \vee (x_2 \wedge x_3) \vee x_4.$$

where x_i can be assign True/False values. Can the formula be set to true by some assignment

Brute force: try all possible assignments 2^n

The proof constructs an instance of the Boolean Satisfiability problem which is satisfiable iff the original instance had a "Yes" answer