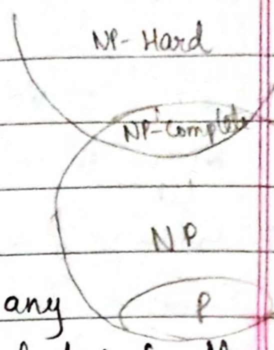
22.04.2020

22. NP-Complete and NP-Hard Problems.

- A problem is in the class NP-Complete (NPC) if it is in NP and is as hard as any problem in NP.

A problem is NP-hard if all the problems in NP are polynomial time reducible to it, even though it may not be in NP itself.

- Every problem in NP can be reduced to another NP problem, in a polynomial way.
- If a problem in NP-Complete class can be solved using a polynomial deterministic algorithm, then all problems in NP can be solved in polynomial time.



Reduction

- A problem 'A' can be reduced to another problem 'B' if any instance of A can be rephrased / transformed into any instance of B, the solution to which provides the solution to the instance of A.

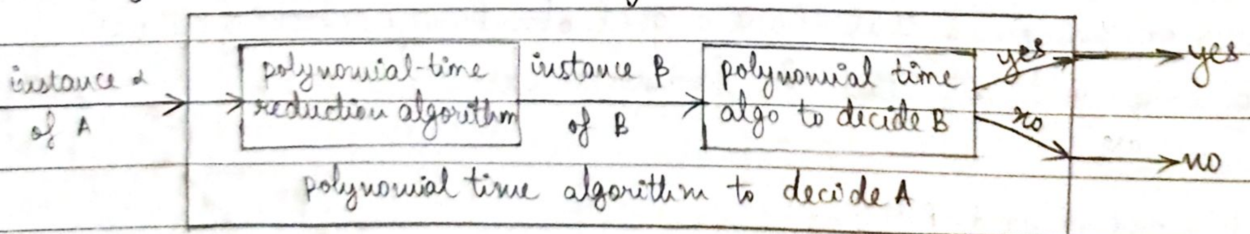
Intuitively: If A reduces in polynomial time to B, A is "no harder to solve" than B.

In other words:

- Suppose we already know how to solve a decision problem B in polynomial time
- Suppose we have a procedure that transforms an instance α of A into some instance β of B with the foll. characteristics.

  (i) The transformation takes polynomial time, ie, there exists a function f that converts the inputs of A to i/ps of B in polynomial time.

  (ii) $A(i) = YES \iff B(i) = YES$, ie, the answers are same; the ans. for α is "yes" iff the ans. for β is also "yes".
- This procedure is called polynomial time reduction algorithm, and we say that A is polynomially reducible to B ($A \leq_p B$)



- It provides us a way to solve problem A in polynomial time:
  1. Given an instance α of A, use polynomial time reduction algo. to transform it to an instance β of problem B.
  2. Run the polynomial time decision algorithm for B on the instance β.
  3. Use the answer for β as the answer for α.

- By "reducing" solving problem A to solving prob. B, we use the "easiness" of B to prove the "easiness" of A.

<u>NP-Completeness (formally)</u>

- A problem B is <u>NP-Complete</u> if:
  (i) $B \in NP$
  (ii) $A \leq_p B$ for all $A \in NP$.
- If B satisfies ~~only~~ property (ii), but not necessarily property (i), B is known as <u>NP-Hard</u>
- Informally, a search problem B is NP-Hard if there exists some NP-complete problem A that turing reduces to B.
- No polynomial time algorithm has been discovered for an NP-Complete problem.
- The problem in NP-Hard cannot be solved until P=NP.

- The problems that cannot be solved by any algorithms are called <u>Undecidable Problems</u>

egs: