- (p. 326) **NP-hard: All problems in NP are polynomial time reducible to it, even though it may not be in NP**.
  - i.e. may be “Harder” than NP – often called “intractable”
  - E.g. D={<p>| p a polynomial in several variables that has an integral root}
    - Show that 3SAT reduces to D
    - Given a wff construct a polynomial with same vars
    - Represent ~x by (1-x), AND by *: (yvz)=(1-(1-y)(1-z)
    - Assignment exists iff 1/0 roots exist
- From https://en.wikipedia.org/wiki/NP-hardness
- **NP**: Class of computational problems for which a given solution can be verified as a solution in polynomial time by a deterministic Turing machine (or solvable by a non-deterministic Turing machine in polynomial time).
- **NP-hard**: Class of problems which are at least as hard as the hardest problems in NP. Problems that are NP-hard do not have to be elements of NP; indeed, they may not even be decidable.
- **NP-complete**: Class of problems which contains the hardest problems in NP. Each NP-complete problem has to be in NP.
- **NP-easy**: At most as hard as NP, but not necessarily in NP, since they may not be decision problems.

- **NP-equivalent**: Problems that are both NP-hard and NP-easy, but not necessarily in NP, since they may not be decision problems.
- **NP-intermediate**: If P and NP are different, then there exist problems in the region of NP that fall between P and the NP-complete problems. (If P and NP are the same class, then NP-intermediate problems do not exist.)

- Summary: from https://people.eecs.berkeley.edu/~vazirani/algorithms/chap8.pdf

Hard problems (NP-complete)	Easy problems (in P)
3SAT	2SAT, HORN SAT
TRAVELING SALESMAN PROBLEM	MINIMUM SPANNING TREE
LONGEST PATH	SHORTEST PATH
3D MATCHING	BIPARTITE MATCHING
KNAPSACK	UNARY KNAPSACK
INDEPENDENT SET	INDEPENDENT SET on trees
INTEGER LINEAR PROGRAMMING	LINEAR PROGRAMMING
RUDRATA PATH	EULER PATH
BALANCED CUT	MINIMUM CUT

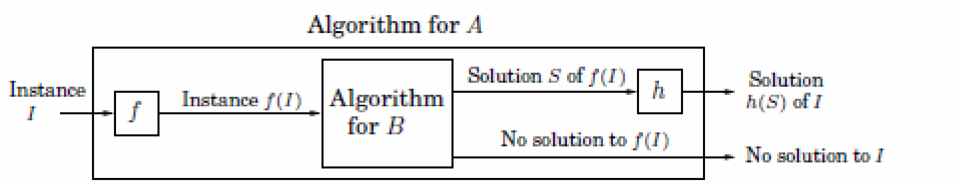

Figure 8.7 Reductions between search problems.