Section 22.1: Reductions Revisited

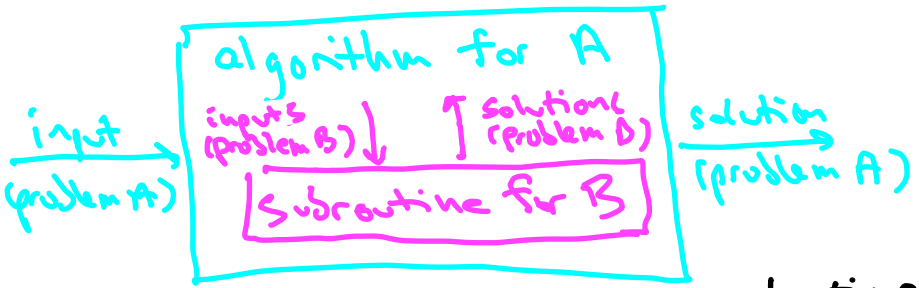
Reductions

NP-hard problem: A polynomial-time algorithm would refute the "P ≠ NP" conjecture ⇒ automatically gives polynomial-time algorithms for thousands of unsolved problems.

[P ≠ NP conjecture ≈ checking validity of a solution can be fundamentally easier than coming up with your own from scratch]

Reduction:

(from A to B)

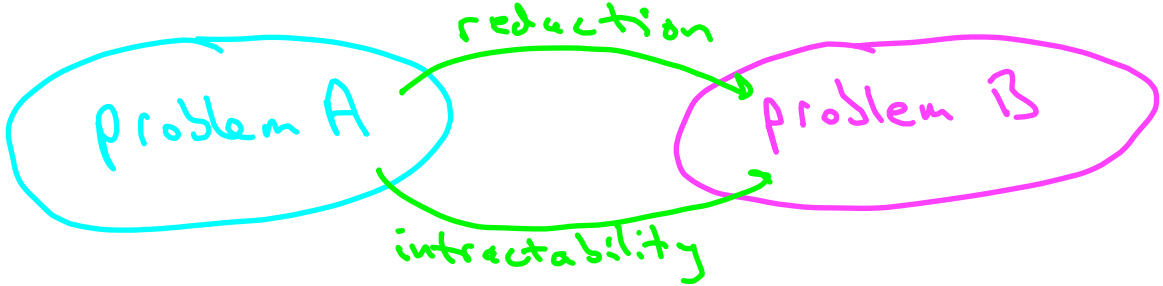


- can invoke subroutine for B a polynomial # of times
- can perform polynomial amount of additional work

Point: spreads poly-time tractability from B to A.

How to Prove a Problem NP-Hard

Reductions spread NP-hardness:



SIMPLE 2-STEP RECIPE

To prove B is NP-hard:

(1) Choose a known NP-hard problem A.

(2) Prove that A reduces to B.