Reductions Redux

Or think of it as a contrapositive:
If we have an algorithm for B, then we also have one for A.
If we don't (expect) to have an algorithm for A, then we don't (expect) to have one for B.

When we reduced A to B before, it was because we had an algorithm for B already, and wanted to solve A.

We knew how to handle unweighted graphs, now we want to see if we can handle weighted.

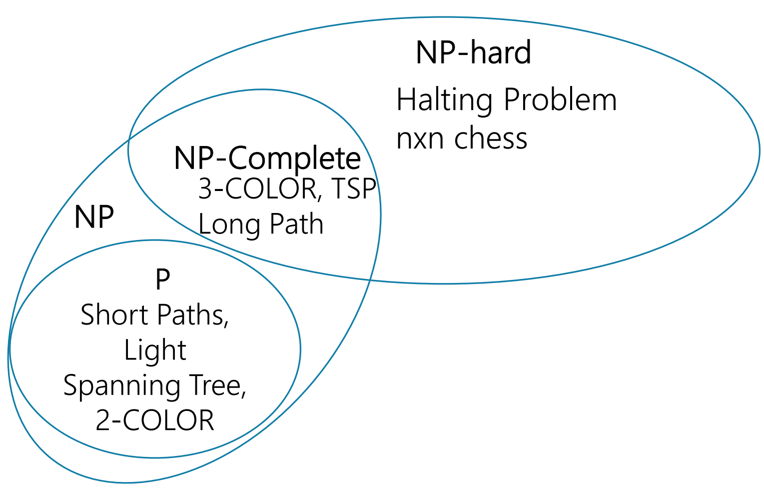
In complexity theory (where we're trying to show algorithms don't exist) we reduce well-studied A to new problem B.

Goal is a proof by contradiction.

1. Suppose (for sake of contradiction) new problem B has a nice algorithm.
2. But then we can use that for an algorithm well-studied problem A.
3. But, uh, no one knows an algorithm for well-studied problem A.
4. “contradiction”

41

What The World Looks Like (We Think)



48

P (stands for "Polynomial")

The set of all decision problems that have an algorithm that runs in time $\boldsymbol{O}(\boldsymbol{n}^{\boldsymbol{k}})$ for some constant k.

NP (stands for "nondeterministic polynomial")

The set of all decision problems such that if the answer is YES, there is a proof of that which can be verified in polynomial time.

NP-complete

Problem B is NP-complete if B is in NP and
for all problems A in NP, A reduces to B in polynomial time.

NP-hard

Problem B is NP-hard if
for all problems A in NP, A reduces to B in polynomial time.

71

Reductions

Polynomial Time Reducible

We say A reduces to B in polynomial time, if there is an algorithm that, using a (hypothetical) polynomial-time algorithm for B, solves problem A in polynomial-time.
Written $A \leq_P B$

If A reduces to B then A should be "easier" than B. (for us as algorithm designers) (thus $A \leq_P B$)

EXP (stands for "Exponential")

The set of all decision problems that have an algorithm that runs in time $\boldsymbol{O}(\boldsymbol{2}^{\boldsymbol{n}^\wedge\boldsymbol{k}})$ for some constant k.

72