Complexity theory

Decision problem – answers yes/no question

Example: "find maximum size of an independent set in a graph" (search problem)
"is there an independent set of the size $\geq k$" (decision problem)

Size of input – number of bits we need to write the problem input
(effectively proportional to the dimension/size of the problem)

Class P – all decision problems that can be solved in running time $p(n)$

polynomial

size of input

Class NP – all decision problems that have a certificate that can be checked in $p(n)$ time

- What is a certificate?
  e.g., "an answer": the vertices that give the biggest independent set
  [it can be verified quickly but not necessarily constructed]
- $P \subset NP$: an algorithm itself is a certificate

Is $P = NP$? Not known (but not likely :))

Assuming $P \neq NP$, here is the notion of hardness we will look for:

a problem is at least as hard as problems known to be in NP-class

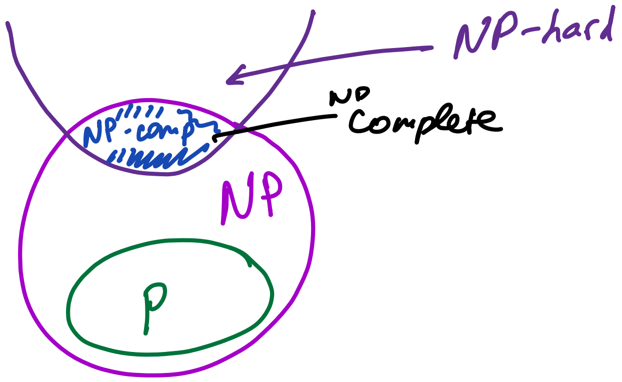
(NP) hard problem

Polynomial time reduction: $A \rightsquigarrow B$ or $A \leq_p B$

A can be reduced to B if arbitrary instances of problem A can be solved using
- polynomial number of steps +
- polynomial number of calls to a problem from B

• A is reducible to B means "A is not harder than B"

Class NP-hard – problem X such as there is a reduction from an NP-hard problem to X.



What does this imply?

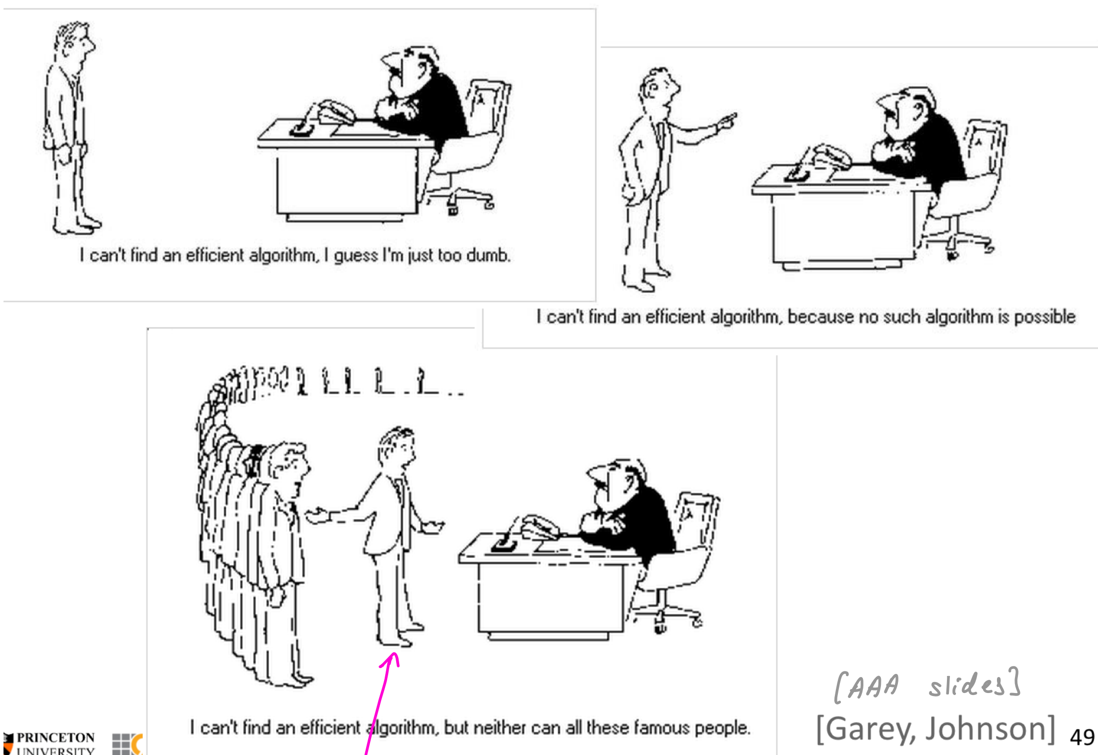
$K \le_p X$

known to be NP-hard (at least as hard as all problems in NP)

comes from the first NP-hard problem

If there is a poly algorithm for X $\Rightarrow$ there is a poly algorithm for any NP-problem $\Rightarrow$ P=NP

The value of reductions



[AAA slides]

[Garey, Johnson]

If we can solve this problem polynomially, then we can solve all NP-problems polynomially (P=NP)

Reductions can be used to show that a problem is "easy" (in Ⓟ)

Example: LP - known to be in Ⓟ

MAX FLOW: input: directed graph with rational weights on edges (capacities) with selected vertices S (sourse) and T (target)

question: is there a flow of value $\geq k$?

[assignments of nonnegative flow values at the edges not above the capacities, so that inflow and outflow is the same for every vertex except S and T]

Claim: MAXFLOW can be formulated as an LP feasibility problem

$$\begin{cases} \sum_{v:\, S \to v} f(S,v) \geq k \\ \sum_{u \to v} f(u,v) = \sum_{v \to w} f(v,w) \\ 0 \leq f(u,v) \leq c(u,v) \end{cases}$$

So, any instance of MAXFLOW is a particular instance of LP

We can solve any MAXFLOW by solving an LP $\Rightarrow$ MAXFLOW $\leq_p$ LP $\Rightarrow$ MAXFLOW is in Ⓟ

(MAXFLOW reduces to LP)

$\text{MINCUT}_{S\text{-}T}$: input: the same

q: is there a partition into 2 sets $\mathcal{S}_1$ and $\mathcal{S}_2$ so that $S \in \mathcal{S}_1$, $T \in \mathcal{S}_2$ so that total capacity of the edges between $\mathcal{S}_1$ and $\mathcal{S}_2$ $\leq k$?

Exercise: • $\text{MINCUT}_{S\text{-}T} \leq_p$ MAXFLOW

• $\text{MINCUT} \leq_p \text{MINCUT}_{S-T}$

(via polynomially many calls to the instance of class MINCUT_{S-T})

Reductions to show that a problem is hard.

Gameplan: find an NP-hard problem (K), so that we can solve (K) by solving instances of (X) poly times ($(K) \leq_p (X)$)

What are the problems from NP-hard?

First examples

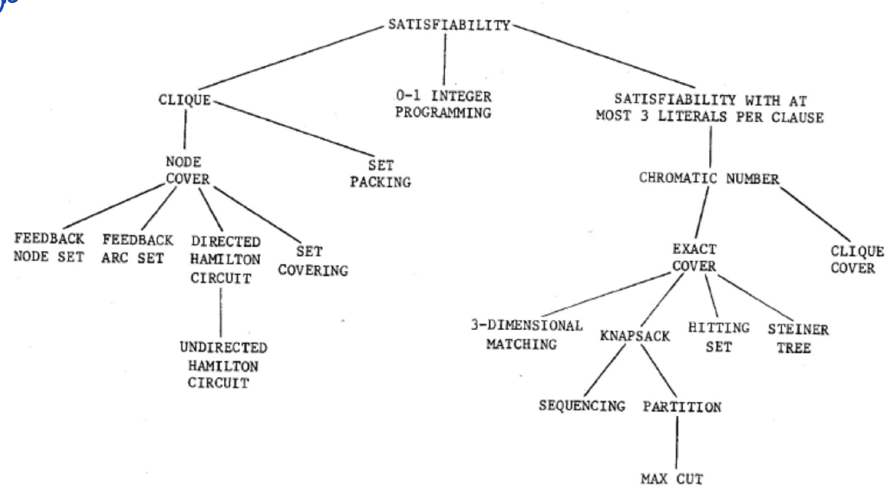


FIGURE 1 - Complete Problems