CSE 417: Algorithms and Computational Complexity

Winter 2001

Lecture 22

Instructor: Paul Beame

1

Polynomial time

- Define P (polynomial-time) to be
  - the set of all decision problems solvable by algorithms whose worst-case running time is bounded by some polynomial in the input size.
- $P = \bigcup_{k\geq 0} \mathrm{TIME}(n^k)$

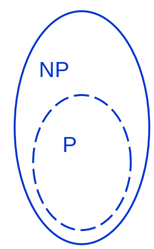
2

The complexity class NP

- NP consists of all decision problems where one can verify the YES answers efficiently (in polynomial time) given a short (polynomial-size) hint.
- Some problems in NP and their hints
  - DecisionTSP: the tour itself,
  - Independent-Set, Clique: the set U
  - Satisfiability: an assignment that makes F true.

3

P and NP



4

NP-hardness & NP-completeness

- Some problems in NP seem hard
  - people have looked for efficient algorithms for them for hundreds of years without success
- However
  - nobody knows how to **prove** that they are really hard to solve, i.e. $P \neq NP$

5

NP-hardness & NP-completeness

- Alternative approach
  - show that they are at least as hard as any problem in NP
- Rough definition:
  - A problem is NP-hard iff it is at least as hard as any problem in NP
  - A problem is NP-complete iff it is both
    - NP-hard
    - in NP

6

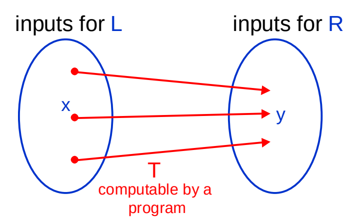
Showing one problem is at least as hard as another

- Reductions
  - Used before in case of considering whether problems could be solve by programs
    - we allowed any old program T as the reduction

Polynomial-time reductions

- We have a problem using our ordinary notion of reduction in complexity
  - We want that if $L \leq R$ then, up to a small amount of slop, L is at least as easy as R
  - But...
    - our reduction T might take a really, really long time even though it is computable by a program
  - Solution:
    - require that program T be efficient, too.

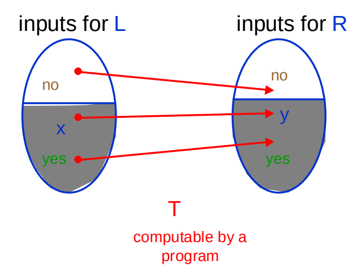
Reduction $L \leq R$



$L(x)=R(T(x))$

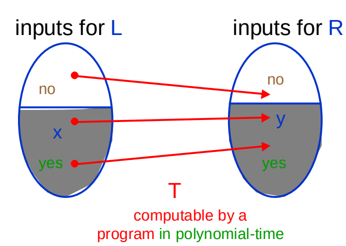
Intuition: L is at least as easy as R or, equivalently, R is at least as hard as L

Decision Problems: Reduction $L \leq R$



L answers yes to x ⇔ R answers yes to T(x)

Polynomial-time reduction $L \leq^p R$



L answers yes to x ⇔ R answers yes to T(x)

Polynomial-time reductions preserve polynomial time

- Theorem: If $L \leq^p R$ and R is in P then L is also in P
- Proof:
  - Let T be reduction showing $L \leq^p R$ that runs in polynomial time, say cn^t.
  - Let A be the algorithm solving R that runs in polynomial time, say dn^r
  - Our algorithm B for L first runs T and then runs A on the output of T

Running time of B

- Running T takes time at most cn^t.
  - The output of T has at most $c'n^t$ bits for some constant c' because we can only create a constant number of output bits per output per step.
- Running A on T's output takes time at most $d(c'n^t)^r$.
  - The input of A is size at most $c'n^t$.
- Total run-time is $O(n^{tr})$ which is polynomial

13

NP-hardness & NP-completeness

- Definition: A problem R is NP-hard iff every problem $L \in NP$ satisfies $L \leq^p R$
- Definition: A problem R is NP-complete iff R is NP-hard and $R \in NP$
- Not obvious that such problems even exist!

14

e.g., Independent-Set $\leq^P$ Clique

- Independent-Set:
  - Given a graph G=(V,E) and an integer k, is there a subset U of V with $|U| \geq k$ such that no two vertices in U are joined by an edge.
- Clique:
  - Given a graph G=(V,E) and an integer k, is there a subset U of V with $|U| \geq k$ such that every pair of vertices in U is joined by an edge.
- What is the reduction T?

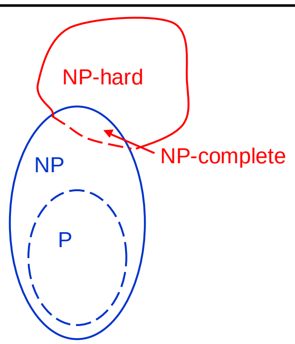
15

Cook's Theorem

- Theorem (Cook 1971): Satisfiability is NP-complete
- Proof idea:
  - Given any problem L in NP
  - Look at the algorithm that verifies the hint for L
  - One can write a polynomial-size CNF formula $F_{L,x}$ that that can be made true iff the verification algorithm for L, on input x and the hint, liked the hint for L
  - $F_{L,x}$ is satisfiable iff L answers YES on input x
  - The transformation from x to $F_{L,x}$ can be computed efficiently

16

P and NP



17