CSE 421 Section 9

NP-completeness

Administrivia

Announcements & Reminders

- **HW6** regrade requests are open

- **HW7**
  - Due **tomorrow** 11/22 @ 11:59pm
  - Late submissions will be open until Sunday, 11/24 @ 11:59pm

- **HW8**
  - Will be released over the weekend
  - Due Wednesday, December 4th @ 11:59pm

- No section next week, happy Thanksgiving!

Definition review

Definition review

There were many new definitions in lecture recently that we'll review now.

To check your understanding, for each definition starting next slide, give an example of the definition!

Definition review

- **Problem**: a set of inputs and the correct outputs
- **Instance**: a single input to a problem
- **Decision problem**: a problem where the output is “yes” or “no”
- **Reduction**:

  $A \leq_p B$ “A reduces to B” “A is not harder than B” “Solve A using B”

  Formally, $A \leq_p B$ if there is an algorithm that solves A using polynomially many calls to a solver for B, running in polynomial time (excluding calls to B).

Definition review

- **P (“polynomial”)**: The set of **decision** problems A that can be solved in poly time.
- **NP (“nondeterministic polynomial”)**: The set of **decision** problems A for which YES-instances can be verified in poly time.

  Formally, there is a poly time algorithm VERIFYA such that for all inputs x,
  - If x is YES, there exists a poly length string y such that $\text{VERIFYA}(x, y) = \text{YES}$.
  - If x is NO, then for all poly length strings y, $\text{VERIFYA}(x, y) = \text{NO}$.
- **NP-hard**: A problem B is NP-hard if $A \leq_p B$ for all A in NP.
- **NP-complete**: A problem B is NP-complete if B is in NP and B is NP-hard.

Definition review

- **Boolean literal**: A Boolean variable x_i or its negation $\neg x_i$
- **Clause**: OR of zero or more literals
- **CNF formula**: AND of zero or more clauses
- **3SAT problem**:

  Input: A CNF formula with exactly 3 literals per clause
  Output: Is there an assignment to the variables that makes the formula true?

  3SAT is a fundamental **NP-complete** problem.

Practice with SAT

Problem 1 – SATisfy This

Determine whether each instance of 3-SAT is satisfiable. If it is, list a satisfying variable assignment.

a) $(\neg a \vee \neg b \vee c) \wedge (a \vee c \vee \neg d) \wedge (b \vee \neg c \vee \neg d) \wedge (\neg a \vee b \vee c) \wedge (\neg b \vee c \vee \neg d)$

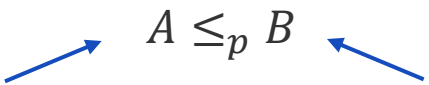
b) $(\neg a \vee b \vee d) \wedge (\neg b \vee c \vee d) \wedge (a \vee \neg c \vee d) \wedge (a \vee \neg b \vee \neg d)$
$\wedge (b \vee \neg c \vee \neg d) \wedge (\neg a \vee c \vee \neg d) \wedge (a \vee b \vee c) \wedge (\neg a \vee \neg b \vee \neg c)$

Think about it with the people around you, then we'll discuss!

Reductions

How to prove NP-hardness

In previous weeks of this class, you've seen reductions of the following form:



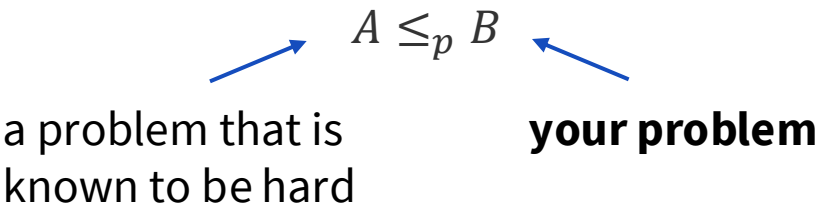
your problem

an algorithm to be used as a library,
i.e. stable matching, network flow

"**My problem is easy** to solve, because I can just use B."

How to prove NP-hardness

Now, for NP-hardness, we need to do the opposite.



"**My problem is hard**, because if it were easy, then A would be easy, but A is hard."

In other words, we convert **from instances of the hard problem to your problem**.

NOT solving your problem!

How to prove NP-completeness

Show B is in NP:

1. State what the certificate is.
2. Say why the certificate can be checked in polynomial time.

Show B is NP-hard:

3. Identify an NP-hard problem A and say, "We will reduce from A to B".
4. Define a reduction function f, which converts instances of A into instances of B.
5. Say why f is computable in polynomial time.
6. Show that "x is a YES-instance for A" $\Rightarrow$ "$f(x)$ is a YES-instance for B".
   - **Convert a certificate** for x into a certificate for $f(x)$.
7. Show that "$f(x)$ is a YES-instance for B" $\Rightarrow$ "x is a YES-instance for A".
   - **Convert a certificate** for $f(x)$ into a certificate for x.

Problem 2 – 5SAT

Define the problem **5SAT** to be:
Input: A CNF formula with exactly 5 literals per clause
Output: Is there an assignment to the variables that makes the formula true?
We will show that 5SAT is NP-complete.

First, we will show that 5SAT is in NP.

a) State what the certificate is.
b) Say why the certificate can be checked in polynomial time.

Think about this briefly!

Problem 2 – 5SAT

Recall **3SAT**:
Input: A CNF formula with exactly 3 literals per clause
Output: Is there an assignment to the variables that makes the formula true?

We will now prove that 5SAT is NP-hard with a reduction involving 3SAT.

c) Fill in the blank: "We will reduce from __ to __". Which is A, and which is B?

Think about this briefly!

Problem 2 – 5SAT

c) Fill in the blank: “We will reduce from __ to __”. Which is A, and which is B?

d) Define a reduction function f, which converts instances of A into B.

Problem 2 – 5SAT

e) Say why f is computable in polynomial time.

Think about this briefly!

Problem 2 – 5SAT

To prove the correctness:

f) Show that "x is a YES-instance for A" ⇒ "$f(x)$ is a YES-instance for B".
(Remember: convert a certificate for x into a certificate for $f(x)$!)

Think about it with the people around you, then we'll discuss!

Problem 2 – 5SAT

To prove the correctness:

g) Show that "$f(x)$ is a YES-instance for B" ⇒ "x is a YES-instance for A".
(Remember: convert a certificate for $f(x)$ into a certificate for x!)

Think about it with the people around you, then we'll discuss!

Problem 3 – Reduction with different types

The Integer Linear Programming problem (**ILP**) is:
Input: An integer matrix A and integer vector b
Output: Is there an integer vector x such that $Ax \leq b$?

Decision version!
Nothing to optimize for.

In lecture, you saw that **3SAT** $\leq_P$ **ILP** via a long series of reductions.
Prove this directly by a single reduction. (We will skip showing ILP is in NP today.)

c) Fill in the blank: "We will reduce from __ to __". Which is A, and which is B?

Think about this briefly!

Problem 3 – Reduction with different types

d) Define a reduction function f, which converts instances of A into B.

(We did an example on the previous slide, so the question is just: how to write this generally?)

Think about it with the people around you, then we'll discuss!

Problem 3 – Reduction with different types

e) Say why f is computable in polynomial time.

Think about this briefly!

Problem 3 – Reduction with different types

To prove the correctness:

f) Show that "x is a YES-instance for A" ⇒ "$f(x)$ is a YES-instance for B".
(Remember: convert a certificate for x into a certificate for $f(x)$!)

Think about it with the people around you, then we'll discuss!

Problem 3 – Reduction with different types

To prove the correctness:

g) Show that "$f(x)$ is a YES-instance for B" ⇒ "x is a YES-instance for A".
(Remember: convert a certificate for $f(x)$ into a certificate for x!)

Think about it with the people around you, then we'll discuss!

Summary

- When you want **to show B is NP-complete, do NOT solve B**!
  - Convert instances of another NP-hard problem A into instances of B.
- For the reduction proof, **convert certificates** of each problem to certificates of the other problem.

Thanks for coming to section this week!