CSE 421 Section 9

P, NP, Reductions

Administrivia

Announcements & Reminders

- HW 6
  - If you think something was graded incorrectly, submit a regrade request!
- HW 7
  - Was due yesterday, Wednesday 11/29
- HW 8
  - It's the last homework, woohoo!
  - Due Wednesday 12/6
- Final Exam
  - Scheduled for **Monday 12/11 @ 2:30-4:20 in our normal room, CSE2 G20**

Problems, P, & NP

First, some Definitions:

- **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**: $A \leq_p B$
  - Informally: **A reduces to B** means **“we can solve A using a library for B”**
  - Formally: A reduces to B in **polynomial time** if there is an algorithm to solve problem A, which, if given access to a library function for solving problem B, calls the library at most polynomially-many times and takes at most polynomial-time otherwise excluding the calls to the library.

P, NP, and P vs. NP

- **P (“polynomial”)**: The set of all decision problems for which there exists an algorithm that runs in time $\mathcal{O}(n^k)$ for some constant k, where n is the size of the input
- **NP (“nondeterministic polynomial”)**: The set of all decision problems such that for every YES-instance (of size n), there is a certificate (of size $\mathcal{O}(n^k)$) for that instance which can be verified in polynomial time
- **P vs. NP**: Are P and NP the same complexity class?
  - That is, can every problem that can be **verified** in polynomial time also be **solved** in polynomial time?

NP-hard, NP-complete

- **NP-hard**: The problem B is NP-hard if for all problems A in NP, A polytime reduces to B

- **NP-complete**: The problem B is NP-complete if B is in NP and B is NP-hard

3SAT

In simple terms:
Input: expression in CNF (AND of ORs) form, where every term has exactly 3 literals involving different variables
Output: true if there is a variable setting which makes the whole expression true, false otherwise.

More formally:
Input:

- A list of Boolean variables $x_1, \ldots, x_n$
- A list of constraints, all of which must be met. Each constraint is of the form: $(z_i \vee z_j \vee z_k)$, where z_i is a "literal" (a variable or the negation of a variable).

Output: true if there is a setting of the variables where all constraints are met, false otherwise.

3SAT

Why is it called 3SAT? 3 because you have 3 variables per constraint, SAT is short for “satisfiability”. The problem is asking, can you find an assignment that satisfies all of the constraints?

3SAT is an **NP-Complete problem**. This means every problem in NP can be reduced to it (it is NP-Hard) and it is also in NP.

1. SATisfy This

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

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

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

Work through (a) and (b) with the people around you, and then we'll go over them together!

Problem 1 – SATisfy This

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

Problem 1 – SATisfy This

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

This is satisfiable.

Problem 1 – SATisfy This

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

This is satisfiable: a = T, b = F, c = T, d = F makes each clause true, so the overall formula is true.

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

$(\neg T \vee \neg F \vee T) \wedge (T \vee T \vee \neg F) \wedge (F \vee \neg T \vee \neg F) \wedge (\neg T \vee F \vee T) \wedge (\neg F \vee T \vee \neg F)$

$(F \vee T \vee T) \wedge (T \vee T \vee T) \wedge (F \vee F \vee T) \wedge (F \vee F \vee T) \wedge (T \vee T \vee T)$

Since every clause has at least one literal that is true, the formula is true!

Problem 1 – SATisfy This

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

Problem 1 – SATisfy This

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

This is NOT satisfiable. There is no assignment that makes every clause true, so the overall formula is false.

Problem 1 – SATisfy This

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

To confirm, we have to check every possibility: a = T, b = T, c = T, d = T

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

$(\neg T \vee T \vee T) \wedge (\neg T \vee T \vee T) \wedge (T \vee \neg T \vee T) \wedge (T \vee \neg T \vee \neg T) \wedge (T \vee \neg T \vee \neg T)$
$\wedge (\neg T \vee T \vee \neg T) \wedge (T \vee T \vee T) \wedge (\neg T \vee \neg T \vee \neg T)$

$(F \vee T \vee T) \wedge (F \vee T \vee T) \wedge (T \vee F \vee T) \wedge (T \vee F \vee F) \wedge (T \vee F \vee F)$
$\wedge (F \vee T \vee F) \wedge (T \vee T \vee T) \wedge (\boldsymbol{F} \vee \boldsymbol{F} \vee \boldsymbol{F})$

Problem 1 – SATisfy This

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

To confirm, we have to check every possibility: a = F, b = T, c = T, d = T

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

$(\neg F \vee T \vee T) \wedge (\neg T \vee T \vee T) \wedge (F \vee \neg T \vee T) \wedge (F \vee \neg T \vee \neg T) \wedge (T \vee \neg T \vee \neg T)$
$\wedge (\neg F \vee T \vee \neg T) \wedge (F \vee T \vee T) \wedge (\neg F \vee \neg T \vee \neg T)$

$(T \vee T \vee T) \wedge (F \vee T \vee T) \wedge (F \vee F \vee T) \wedge (\boldsymbol{F} \vee \boldsymbol{F} \vee \boldsymbol{F}) \wedge (T \vee F \vee F)$
$\wedge (T \vee T \vee F) \wedge (F \vee T \vee T) \wedge (T \vee F \vee F)$

Problem 1 – SATisfy This

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

To confirm, we have to check every possibility: a = T, b = F, c = T, d = T

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

$(\neg T \vee F \vee T) \wedge (\neg F \vee T \vee T) \wedge (T \vee \neg T \vee T) \wedge (T \vee \neg F \vee \neg T) \wedge (F \vee \neg T \vee \neg T)$
$\wedge (\neg T \vee T \vee \neg T) \wedge (T \vee F \vee T) \wedge (\neg T \vee \neg F \vee \neg T)$

$(F \vee F \vee T) \wedge (T \vee T \vee T) \wedge (T \vee F \vee T) \wedge (T \vee T \vee F) \wedge (\boldsymbol{F} \vee \boldsymbol{F} \vee \boldsymbol{F})$
$\wedge (F \vee T \vee F) \wedge (T \vee F \vee T) \wedge (F \vee T \vee F)$

Problem 1 – SATisfy This

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

To confirm, we have to check every possibility: a = T, b = T, c = F, d = T

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

$(\neg T \vee T \vee T) \wedge (\neg T \vee F \vee T) \wedge (T \vee \neg F \vee T) \wedge (T \vee \neg T \vee \neg T) \wedge (T \vee \neg F \vee \neg T)$
$\wedge (\neg T \vee F \vee \neg T) \wedge (T \vee T \vee F) \wedge (\neg T \vee \neg T \vee \neg F)$

$(F \vee T \vee T) \wedge (F \vee F \vee T) \wedge (T \vee T \vee T) \wedge (T \vee F \vee F) \wedge (T \vee T \vee F)$
$\wedge (\boldsymbol{F} \vee \boldsymbol{F} \vee \boldsymbol{F}) \wedge (T \vee T \vee F) \wedge (F \vee F \vee T)$

Problem 1 – SATisfy This

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

To confirm, we have to check every possibility: a = T, b = T, c = T, d = F

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

$(\neg T \vee T \vee F) \wedge (\neg T \vee T \vee F) \wedge (T \vee \neg T \vee F) \wedge (T \vee \neg T \vee \neg F) \wedge (T \vee \neg T \vee \neg F)$
$\wedge (\neg T \vee T \vee \neg F) \wedge (T \vee T \vee T) \wedge (\neg T \vee \neg T \vee \neg T)$

$(F \vee T \vee F) \wedge (F \vee T \vee F) \wedge (T \vee F \vee F) \wedge (T \vee F \vee T) \wedge (T \vee F \vee T)$
$\wedge (F \vee T \vee T) \wedge (T \vee T \vee T) \wedge (\boldsymbol{F} \vee \boldsymbol{F} \vee \boldsymbol{F})$

Problem 1 – SATisfy This

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

To confirm, we have to check every possibility: a = F, b = F, c = T, d = T

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

$(\neg F \vee F \vee T) \wedge (\neg F \vee T \vee T) \wedge (F \vee \neg T \vee T) \wedge (F \vee \neg F \vee \neg T) \wedge (F \vee \neg T \vee \neg T)$
$\wedge (\neg F \vee T \vee \neg T) \wedge (F \vee F \vee T) \wedge (\neg F \vee \neg F \vee \neg T)$

$(T \vee F \vee T) \wedge (T \vee T \vee T) \wedge (F \vee F \vee T) \wedge (F \vee T \vee F) \wedge (\boldsymbol{F} \vee \boldsymbol{F} \vee \boldsymbol{F})$
$\wedge (T \vee T \vee F) \wedge (F \vee F \vee T) \wedge (T \vee T \vee F)$

Problem 1 – SATisfy This

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

To confirm, we have to check every possibility: a = F, b = T, c = F, d = T

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

$(\neg F \vee T \vee T) \wedge (\neg T \vee F \vee T) \wedge (F \vee \neg F \vee T) \wedge (F \vee \neg T \vee \neg T) \wedge (T \vee \neg F \vee \neg T)$
$\wedge (\neg F \vee F \vee \neg T) \wedge (F \vee T \vee F) \wedge (\neg F \vee \neg T \vee \neg F)$

$(T \vee T \vee T) \wedge (F \vee F \vee T) \wedge (F \vee T \vee T) \wedge (\boldsymbol{F} \vee \boldsymbol{F} \vee \boldsymbol{F}) \wedge (T \vee T \vee F)$
$\wedge (T \vee F \vee F) \wedge (F \vee T \vee F) \wedge (T \vee F \vee T)$

Problem 1 – SATisfy This

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

To confirm, we have to check every possibility: a = F, b = T, c = T, d = F

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

$(\neg F \vee T \vee F) \wedge (\neg T \vee T \vee F) \wedge (F \vee \neg T \vee F) \wedge (F \vee \neg T \vee \neg F) \wedge (T \vee \neg T \vee \neg F)$
$\wedge (\neg F \vee T \vee \neg F) \wedge (F \vee T \vee T) \wedge (\neg F \vee \neg T \vee \neg T)$

$(T \vee T \vee F) \wedge (F \vee T \vee F) \wedge (\boldsymbol{F} \vee \boldsymbol{F} \vee \boldsymbol{F}) \wedge (F \vee F \vee T) \wedge (T \vee F \vee T)$
$\wedge (T \vee T \vee T) \wedge (F \vee T \vee T) \wedge (T \vee F \vee F)$

Problem 1 – SATisfy This

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

To confirm, we have to check every possibility: a = T, b = F, c = F, d = T

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

$(\neg T \vee F \vee T) \wedge (\neg F \vee F \vee T) \wedge (T \vee \neg F \vee T) \wedge (T \vee \neg F \vee \neg T) \wedge (F \vee \neg F \vee \neg T)$
$\wedge (\neg T \vee F \vee \neg T) \wedge (T \vee F \vee F) \wedge (\neg T \vee \neg F \vee \neg F)$

$(F \vee F \vee T) \wedge (T \vee F \vee T) \wedge (T \vee T \vee T) \wedge (T \vee T \vee F) \wedge (F \vee T \vee F)$
$\wedge (\boldsymbol{F} \vee \boldsymbol{F} \vee \boldsymbol{F}) \wedge (T \vee F \vee F) \wedge (F \vee T \vee T)$

Problem 1 – SATisfy This

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

To confirm, we have to check every possibility: a = T, b = F, c = T, d = F

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

$(\neg T \vee F \vee F) \wedge (\neg F \vee T \vee F) \wedge (T \vee \neg T \vee F) \wedge (T \vee \neg F \vee \neg F) \wedge (F \vee \neg T \vee \neg F)$
$\wedge (\neg T \vee T \vee \neg F) \wedge (T \vee F \vee T) \wedge (\neg T \vee \neg F \vee \neg T)$

$(\boldsymbol{F} \vee \boldsymbol{F} \vee \boldsymbol{F}) \wedge (T \vee T \vee F) \wedge (T \vee F \vee F) \wedge (T \vee T \vee T) \wedge (F \vee F \vee T)$
$\wedge (F \vee T \vee T) \wedge (T \vee F \vee T) \wedge (F \vee T \vee F)$

Problem 1 – SATisfy This

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

To confirm, we have to check every possibility: a = T, b = T, c = F, d = F

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

$(\neg T \vee T \vee F) \wedge (\neg T \vee F \vee F) \wedge (T \vee \neg F \vee F) \wedge (T \vee \neg T \vee \neg F) \wedge (T \vee \neg F \vee \neg F)$
$\wedge (\neg T \vee F \vee \neg F) \wedge (T \vee T \vee F) \wedge (\neg T \vee \neg T \vee \neg F)$

$(F \vee T \vee F) \wedge (\boldsymbol{F} \vee \boldsymbol{F} \vee \boldsymbol{F}) \wedge (T \vee T \vee F) \wedge (T \vee F \vee T) \wedge (T \vee T \vee T)$
$\wedge (F \vee F \vee T) \wedge (T \vee T \vee F) \wedge (F \vee F \vee T)$

Problem 1 – SATisfy This

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

To confirm, we have to check every possibility: a = F, b = F, c = F, d = T

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

$(\neg F \vee F \vee T) \wedge (\neg F \vee F \vee T) \wedge (F \vee \neg F \vee T) \wedge (F \vee \neg F \vee \neg T) \wedge (F \vee \neg F \vee \neg T)$
$\wedge (\neg F \vee F \vee \neg T) \wedge (F \vee F \vee F) \wedge (\neg F \vee \neg F \vee \neg F)$

$(T \vee F \vee T) \wedge (T \vee F \vee T) \wedge (F \vee T \vee T) \wedge (F \vee T \vee F) \wedge (F \vee T \vee F)$
$\wedge (T \vee F \vee F) \wedge (\boldsymbol{F} \vee \boldsymbol{F} \vee \boldsymbol{F}) \wedge (T \vee T \vee T)$

Problem 1 – SATisfy This

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

To confirm, we have to check every possibility: a = F, b = F, c = T, d = F

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

$(\neg F \vee F \vee F) \wedge (\neg F \vee T \vee F) \wedge (F \vee \neg T \vee F) \wedge (F \vee \neg F \vee \neg F) \wedge (F \vee \neg T \vee \neg F)$
$\wedge (\neg F \vee T \vee \neg F) \wedge (F \vee F \vee T) \wedge (\neg F \vee \neg F \vee \neg T)$

$(T \vee F \vee F) \wedge (T \vee T \vee F) \wedge (\boldsymbol{F} \vee \boldsymbol{F} \vee \boldsymbol{F}) \wedge (F \vee T \vee T) \wedge (F \vee F \vee T)$
$\wedge (T \vee T \vee T) \wedge (F \vee F \vee T) \wedge (T \vee T \vee F)$

Problem 1 – SATisfy This

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

To confirm, we have to check every possibility: a = F, b = T, c = F, d = F

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

$(\neg F \vee T \vee F) \wedge (\neg T \vee F \vee F) \wedge (F \vee \neg F \vee F) \wedge (F \vee \neg T \vee \neg F) \wedge (T \vee \neg F \vee \neg F)$
$\wedge (\neg F \vee F \vee \neg F) \wedge (F \vee T \vee F) \wedge (\neg F \vee \neg T \vee \neg F)$

$(T \vee T \vee F) \wedge (\boldsymbol{F} \vee \boldsymbol{F} \vee \boldsymbol{F}) \wedge (F \vee T \vee F) \wedge (F \vee F \vee T) \wedge (T \vee T \vee T)$
$\wedge (T \vee F \vee T) \wedge (F \vee T \vee F) \wedge (T \vee F \vee T)$

Problem 1 – SATisfy This

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

To confirm, we have to check every possibility: a = T, b = F, c = F, d = F

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

$(\neg T \vee F \vee F) \wedge (\neg F \vee F \vee F) \wedge (T \vee \neg F \vee F) \wedge (T \vee \neg F \vee \neg F) \wedge (F \vee \neg F \vee \neg F)$
$\wedge (\neg T \vee F \vee \neg F) \wedge (T \vee F \vee F) \wedge (\neg T \vee \neg F \vee \neg F)$

$(\boldsymbol{F} \vee \boldsymbol{F} \vee \boldsymbol{F}) \wedge (T \vee F \vee F) \wedge (T \vee T \vee F) \wedge (T \vee T \vee T) \wedge (F \vee T \vee T)$
$\wedge (F \vee F \vee T) \wedge (T \vee F \vee F) \wedge (F \vee T \vee T)$

Problem 1 – SATisfy This

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

To confirm, we have to check every possibility: a = F, b = F, c = F, d = F

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

$(\neg F \vee F \vee F) \wedge (\neg F \vee F \vee F) \wedge (F \vee \neg F \vee F) \wedge (F \vee \neg F \vee \neg F) \wedge (F \vee \neg F \vee \neg F)$
$\wedge (\neg F \vee F \vee \neg F) \wedge (F \vee F \vee F) \wedge (\neg F \vee \neg F \vee \neg F)$

$(T \vee F \vee F) \wedge (T \vee F \vee F) \wedge (F \vee T \vee F) \wedge (F \vee T \vee T) \wedge (F \vee T \vee T)$
$\wedge (T \vee F \vee T) \wedge (\boldsymbol{F} \vee \boldsymbol{F} \vee \boldsymbol{F}) \wedge (T \vee T \vee T)$

Reductions

Why do we care about NP-Hard & NP-Complete?

Let B be an NP-hard problem. Remember, this means that every problem in NP polytime reduces to B. Suppose you found a polynomial time algorithm for B; you now have for free a polynomial time algorithm for every problem in NP, so $P = NP$.

On the other hand, if any problem in NP is not in P (any doesn't have a polynomial time algorithm), then no NP-complete problem is in P.

What can we do with NP-Hard & NP-Complete?

We're pretty sure that there aren't any efficient algorithms for NP-complete problems. When you're asked to write an algorithm for a problem, it is worthwhile for you to be able to tell if the problem is NP-complete or not. How can we do that?

We need to show that a problem that is NP-complete is easier (or at most equal in difficulty) to the problem! If it's harder than an NP-complete problem, then it's NP-hard. If the problem is also in NP, then it's NP-complete as well!

How can we show this? We can write a reduction!

NP-Completeness Reductions

Given problem A, to prove that it is NP-hard, we take a known NP-complete problem B, and show that $B \leq_P A$. To prove that A is NP-complete, we also show that it is in NP.

Essentially, you need to take the input to the NP-complete problem B and transform it into the input to the problem A so that an algorithm for A would return true on this modified input if and only if B is true on the original input.

The reduction algorithm is the process of transforming that input to B into the input to A.

NP-Completeness Reductions: which way?

How do you remember which direction? The core idea of an NP-completeness reduction is a proof by contradiction:

Suppose, for the sake of contradiction, there were a polynomial time algorithm for B. But then if there were, I could use that to design a polynomial time algorithm for problem A. But we *really* don't think there's a polynomial time algorithm for problem A. So we should *really* think there isn't one for B either!

Key Idea: Reduce FROM the known hard problem TO the new problem.

Steps to Proving Problem B is NP-complete

- Show B is in NP
  a) State what the hint/certificate is.
  b) Argue that it is polynomial-time to check.
- Show B is NP-hard:

  State: "Reduction is from NP-hard Problem A"
  a) Show what the reduction function f is.
  b) Argue that f is polynomial time.

  Argue correctness in two directions:
  c) x a YES for $A \Rightarrow f(x)$ is a YES for B
     - Do this by showing how to convert a certificate for x being YES for A to a certificate for $f(x)$ being a YES for B.
  d) $f(x)$ a YES for $B \Rightarrow x$ is a YES for A
     - … by converting certificates for $f(x)$ to certificates for x

Strategy for Reductions

1. Read and Understand the Problem
2. Design the Reduction
3. Write the Proof
    - Prove Run-Time
    - Prove correctness; requires TWO implications:
        - If the correct answer is YES, then our algorithm says YES
        - If our algorithm says YES, then the correct answer is YES

2. A Fun Reduction

Problem 2 – A Fun Reduction

Define 5SAT as the following problem:
Input: An expression in CNF form, where every term has exactly 5 literals on different variables.
Output: `true` if there is a variable setting which makes the whole expression true, `false` otherwise.

And 3SAT as earlier:
Input: expression in CNF form, where every term has 3 literals on different variables.
Output: true if there is a variable setting which makes the whole expression true, false otherwise.

Prove that 5SAT is NP-complete using 3SAT.

Problem 2.1 – Read and Understand the Problem

First understand 5SAT:

- What is the input type?
- What is the output type?
- Are there any technical terms in the problem you don't know? Are there any words that look like normal words, but are actually technical terms?

Then think about the reduction:

- Which problem are you solving, and which problem are you assuming you have an algorithm for? Make sure your reduction is "going the right direction"
- What is the output type for your reduction?

Problem 2.1 – Read and Understand the Problem

First understand 5SAT:

- What is the input type? an expression in CNF form where each clause has 5 literals
- What is the output type?
- Are there any technical terms in the problem you don't know? Are there any words that look like normal words, but are actually technical terms?

Then think about the reduction:

- Which problem are you solving, and which problem are you assuming you have an algorithm for? Make sure your reduction is "going the right direction"

- What is the output type for your reduction?

Problem 2.1 – Read and Understand the Problem

First understand 5SAT:

- What is the input type? an expression in CNF form where each clause has 5 literals
- What is the output type? true or false
- Are there any technical terms in the problem you don't know? Are there any words that look like normal words, but are actually technical terms?

Then think about the reduction:

- Which problem are you solving, and which problem are you assuming you have an algorithm for? Make sure your reduction is "going the right direction"

- What is the output type for your reduction?

Problem 2.1 – Read and Understand the Problem

First understand 5SAT:

- What is the input type? an expression in CNF form where each clause has 5 literals
- What is the output type? true or false
- Are there any technical terms in the problem you don't know? Are there any words that look like normal words, but are actually technical terms?

CNF form is AND of ORs like $(z_a \vee z_d \vee z_f \vee z_h \vee z_i) \wedge \ldots \wedge (z_c \vee z_i \vee z_j \vee z_m \vee z_p)$
literals z_i are boolean variables or the negation of boolean variables x_i or $\neg x_i$

Then think about the reduction:

- Which problem are you solving, and which problem are you assuming you have an algorithm for? Make sure your reduction is "going the right direction"

- What is the output type for your reduction?

Problem 2.1 – Read and Understand the Problem

First understand 5SAT:

- What is the input type? an expression in CNF form where each clause has 5 literals
- What is the output type? true or false
- Are there any technical terms in the problem you don't know? Are there any words that look like normal words, but are actually technical terms?

  CNF form is AND of ORs like $(z_a \vee z_d \vee z_f \vee z_h \vee z_i) \wedge \ldots \wedge (z_c \vee z_i \vee z_j \vee z_m \vee z_p)$
  literals z_i are boolean variables or the negation of boolean variables x_i or $\neg x_i$

Then think about the reduction:

- Which problem are you solving, and which problem are you assuming you have an algorithm for? Make sure your reduction is "going the right direction"

  We need to reduce an NP-complete problem to 5SAT in polynomial time. Assume we have an algorithm for 5SAT. We want to solve 3SAT. In other words, we want to show that 3SAT $\leq_P$ 5SAT.
- What is the output type for your reduction?

Problem 2.1 – Read and Understand the Problem

First understand 5SAT:

- What is the input type? an expression in CNF form where each clause has 5 literals
- What is the output type? true or false
- Are there any technical terms in the problem you don't know? Are there any words that look like normal words, but are actually technical terms?

  CNF form is AND of ORs like $(z_a \vee z_d \vee z_f \vee z_h \vee z_i) \wedge \ldots \wedge (z_c \vee z_i \vee z_j \vee z_m \vee z_p)$
  literals z_i are boolean variables or the negation of boolean variables x_i or $\neg x_i$

Then think about the reduction:

- Which problem are you solving, and which problem are you assuming you have an algorithm for? Make sure your reduction is "going the right direction"

  We need to reduce an NP-complete problem to 5SAT in polynomial time. Assume we have an algorithm for 5SAT. We want to solve 3SAT. In other words, we want to show that 3SAT $\leq_P$ 5SAT.
- What is the output type for your reduction?

  a Boolean which is the answer to the 3SAT (which we get by calling 5SAT like a library function)

Problem 2.2 – Design the Reduction

Now write a reduction. Remember a reduction is an algorithm! It often helps to think about the “certificates” (the thing that makes it a YES instance) and transform from one type of certificate to the other.

Problem 2.2 – Design the Reduction

Now write a reduction. Remember a reduction is an algorithm! It often helps to think about the “certificates” (the thing that makes it a YES instance) and transform from one type of certificate to the other.

Hint: We want to start with the input to 3SAT and transform it into the input to 5SAT so that 5SAT returns true iff 3SAT would also return true!

Problem 2.2 – Design the Reduction

Now write a reduction. Remember a reduction is an algorithm! It often helps to think about the "certificates" (the thing that makes it a YES instance) and transform from one type of certificate to the other.

Let $x_1, \ldots x_n$ be the variables in the 3SAT instance and $C_1, C_2, \ldots, C_m$ be the clauses.

Create two dummy variables d_1, d_2. For each clause C_i, create four clauses:
$C_i \vee d_1 \vee d_2$
$C_i \vee \neg d_1 \vee d_2$
$C_i \vee d_1 \vee \neg d_2$
$C_i \vee \neg\ d_1 \vee \neg d_2$

Our 5SAT instance is: $x_1, \ldots x_n, d_1, d_2$
The $4m$ clauses described above.

Problem 2.3 – Write the Proof

a) To be NP-Complete, 5SAT needs to be in NP. Argue that it is (this argument is usually only 2-3 sentences).

b) Show your reduction is correct. Remember you need to prove two implications and that the running time is polynomial.

Problem 2.3 – Write the Proof

a) To be NP-Complete, 5SAT needs to be in NP. Argue that it is (this argument is usually only 2-3 sentences).

Problem 2.3 – Write the Proof

a) To be NP-Complete, 5SAT needs to be in NP. Argue that it is (this argument is usually only 2-3 sentences).

A verifier would take in the settings of the variables to true and false. Given a setting, a verifier would check that each clause (i.e., each constraint) is satisfied. This will take time linear in the length of the constraints, so it is polynomial time.

Problem 2.3 – Write the Proof

b) Show your reduction is correct. Running Time:

Problem 2.3 – Write the Proof

b) Show your reduction is correct. Running Time:

Running Time: Our algorithm makes 4 copies of every clause and adds a constant length set of literals to each clause, so the running time to create the instance is polynomial (and we call the library only once, which is also at most polynomial).

Problem 2.3 – Write the Proof

b) Show your reduction is correct. Suppose correct answer is YES and our reduction returns YES:

Problem 2.3 – Write the Proof

b) Show your reduction is correct. Suppose correct answer is YES and our reduction returns YES:

Let φ_3 be our 3SAT instance and φ_5 be our 5-SAT instance.

Suppose φ_3 is satisfiable, we show that our reduction returns true. Since φ_3 is satisfiable, there is a setting of the variables which causes φ_3 to be true. Take that setting, and set d_1, d_2 arbitrarily. Every clause of φ_5 is a clause of φ_3 with extra literals ORed on, so since each clause of φ_3 is true, each clause of φ_5 is as well, and this is a satisfying assignment.

Problem 2.3 – Write the Proof

b) Show your reduction is correct. Suppose our reduction returns YES and correct answer is YES:

Problem 2.3 – Write the Proof

b) Show your reduction is correct. Suppose our reduction returns YES and correct answer is YES:

Conversely, suppose that our reduction returns true, and therefore φ_5 was satisfiable. Consider a satisfying assignment for φ_5. We claim that (ignoring d_1, d_2) the same assignment satisfies φ_3. Consider an arbitrary clause C_i of φ_3. In φ_5 there were four clauses built from C_i (each ORed with all combinations of literals of d_1, d_2. One of the created clauses in φ_5 had both inserted literals involving d_1, d_2 being false (since we included all possible combinations). Since φ_5 was satisfied, this clause evaluated to true, which means that C_i evaluated to true. Since C_i was arbitrary, we have that every clause is true, and therefore a satisfying assignment for φ_3, as required.

2. A Reduction with different types

Problem 3 – A Reduction with different types

Define `Integer-Programming (ILP)` as follows:
Input: An integer matrix A and integer vector b
Output: true if there is an integer vector x such that $Ax \leq b,$ false otherwise.

And 3SAT as earlier:
Input: expression in CNF form, where every term has 3 literals on different variables.
Output: true if there is a variable setting which makes the whole expression true, false otherwise.

We already know from class that 3SAT $\leq_P$ ILP by a long series of reductions. Prove this directly by a single reduction.

Problem 3.1 – Read and Understand the Problem

First understand `ILP`:

- What is the input type?
- What is the output type?
- Are there any technical terms in the problem you don't know? Are there any words that look like normal words, but are actually technical terms?

Then think about the reduction:

- Which problem are you solving, and which problem are you assuming you have an algorithm for? Make sure your reduction is "going the right direction"
- What is the output type for your reduction?

Work through these questions with the people around you, and then we'll go over them together!

Problem 3.1 – Read and Understand the Problem

First understand `ILP`:

- What is the input type? an integer matrix and an integer vector
- What is the output type?
- Are there any technical terms in the problem you don't know? Are there any words that look like normal words, but are actually technical terms?

Then think about the reduction:

- Which problem are you solving, and which problem are you assuming you have an algorithm for? Make sure your reduction is "going the right direction"

- What is the output type for your reduction?

Problem 3.1 – Read and Understand the Problem

First understand `ILP`:

- What is the input type? an integer matrix and an integer vector
- What is the output type? true or false
- Are there any technical terms in the problem you don't know? Are there any words that look like normal words, but are actually technical terms?

Then think about the reduction:

- Which problem are you solving, and which problem are you assuming you have an algorithm for? Make sure your reduction is "going the right direction"

- What is the output type for your reduction?

Problem 3.1 – Read and Understand the Problem

First understand `ILP`:

- What is the input type? an integer matrix and an integer vector
- What is the output type? true or false
- Are there any technical terms in the problem you don't know? Are there any words that look like normal words, but are actually technical terms?

  No

Then think about the reduction:

- Which problem are you solving, and which problem are you assuming you have an algorithm for? Make sure your reduction is "going the right direction"

- What is the output type for your reduction?

Problem 3.1 – Read and Understand the Problem

First understand ILP:

- What is the input type? an integer matrix and an integer vector
- What is the output type? true or false
- Are there any technical terms in the problem you don't know? Are there any words that look like normal words, but are actually technical terms?

  No

Then think about the reduction:

- Which problem are you solving, and which problem are you assuming you have an algorithm for? Make sure your reduction is "going the right direction"

  We want to show that 3SAT $\leq_P$ ILP. Assume we have an algorithm for ILP. We're trying to solve 3SAT

- What is the output type for your reduction?

Problem 3.1 – Read and Understand the Problem

First understand `ILP`:

- What is the input type? an integer matrix and an integer vector
- What is the output type? true or false
- Are there any technical terms in the problem you don't know? Are there any words that look like normal words, but are actually technical terms?

  No

Then think about the reduction:

- Which problem are you solving, and which problem are you assuming you have an algorithm for? Make sure your reduction is "going the right direction"

  We want to show that 3SAT $\leq_P$ ILP. Assume we have an algorithm for ILP. We're trying to solve 3SAT

- What is the output type for your reduction?

  a Boolean which is the answer to the 3SAT (which we get by calling ILP like a library function)

Problem 3.2 – Design the Reduction

Now write a reduction. Remember a reduction is an algorithm! It often helps to think about the “certificates” (the thing that makes it a YES instance) and transform from one type of certificate to the other.

Work through this problem with the people around you, and then we’ll go over it together!

Problem 3.2 – Design the Reduction

Now write a reduction. Remember a reduction is an algorithm! It often helps to think about the “certificates” (the thing that makes it a YES instance) and transform from one type of certificate to the other.

Hint: We want to start with the input to 3SAT and transform it into the input to ILP so that ILP returns true iff 3SAT would also return true!

Work through this problem with the people around you, and then we’ll go over it together!

Problem 3.2 – Design the Reduction

Now write a reduction. Remember a reduction is an algorithm! It often helps to think about the “certificates” (the thing that makes it a YES instance) and transform from one type of certificate to the other.

Problem 3.2 – Design the Reduction

Now write a reduction. Remember a reduction is an algorithm! It often helps to think about the "certificates" (the thing that makes it a YES instance) and transform from one type of certificate to the other.

Let's try an example (don't write this for your solution):

$$(\neg w \vee \neg x \vee y) \wedge (w \vee y \vee \neg z) \wedge (w \vee \neg y \vee \neg z) \wedge (w \vee x \vee y)$$

We need to use the ILP to handle two things: The Boolean part of 3SAT and the clause constraints.

Let's start with the Boolean part. It seems natural to have the ILP have a variable for each variable for 3SAT. To make sure that these variables are Boolean, we add the constraints:

$$w \leq 1, x \leq 1, y \leq 1, z \leq 1 \text{ (and } w \geq 0, x \geq 0, y \geq 0, z \geq 0).$$

Problem 3.2 – Design the Reduction

Now write a reduction. Remember a reduction is an algorithm! It often helps to think about the "certificates" (the thing that makes it a YES instance) and transform from one type of certificate to the other.

Let's try an example (don't write this for your solution):

$$(\neg w \vee \neg x \vee y) \wedge (w \vee y \vee \neg z) \wedge (w \vee \neg y \vee \neg z) \wedge (w \vee x \vee y)$$

Let's start with the Boolean part. It seems natural to have the ILP have a variable for each variable for 3SAT. To make sure that these variables are Boolean, we add the constraints:

$$w \leq 1, x \leq 1, y \leq 1, z \leq 1 \text{ (and } w \geq 0, x \geq 0, y \geq 0, z \geq 0).$$

We now need the clause part: To represent the value of the negation of w we can write $1 - w$.
To represent $(\neg w \vee \neg x \vee y)$ we want to say that at least 1 of the literals is true so
we add the constraint $1 - w + 1 - x + y \geq 1$, which we rearrange in standard form as $w + x - y \leq 1$.
Similarly for the other clauses we get $w + y + 1 - z \geq 1$ which rearranges to $z - w - y \leq 0$,
$w + 1 - y + 1 - z \geq 1$ which rearranges to $-w + y + z \leq 1$, and
$w + x + y \geq 1$ which rearranges to $-w - x - y \leq -1$.

Problem 3.2 – Design the Reduction

Now write a reduction. Remember a reduction is an algorithm! It often helps to think about the "certificates" (the thing that makes it a YES instance) and transform from one type of certificate to the other.

We now write the general form.
Let $x_1, \dots x_n$ be the variables in the 3SAT instance and $C_1, C_2, \dots, C_m$ be the clauses.

We add the constraints: $x_i \leq 1$ for all $i = 1, \dots, n$
And we always have: $x_i \geq 0$ for all $i = 1, \dots, n$

For each clause C_j, if the variables are $x_{i_1}, x_{i_2}, x_{i_3}$, include the constraint:

$$\sum_{k=1,2,3} \begin{cases} x_{i_k} \text{ if the literal } x_{i_k} \text{ appears in } C_j \\ 1 - x_{i_k} \text{ if the literal } \neg x_{i_k} \text{ appears in } C_j \end{cases} \geq 1$$

which is equivalent to

$$\sum_{k=1,2,3} \begin{cases} -x_{i_k} \text{ if the literal } x_{i_k} \text{ appears in } C_j \\ x_{i_k} \text{ if the literal } \neg x_{i_k} \text{ appears in } C_j \end{cases} \leq -1 + \text{number of negative literals}$$

in standard form.

Problem 3.3 – Write the Proof

Show your reduction is correct. Remember you need to prove two implications and that the running time is polynomial.

Work through this problem with the people around you, and then we'll go over it together!

Problem 3.3 – Write the Proof

b) Running Time:

Problem 3.3 – Write the Proof

b) Running Time:

Running Time: The reduction creates one equation for every clause, so it is definitely polynomial time.

Problem 3.3 – Write the Proof

c) Show your reduction is correct (⇒). Suppose correct answer is YES, want to show our reduction returns YES:

Problem 3.3 – Write the Proof

c) Show your reduction is correct (⇒). Suppose correct answer is YES, want to show our reduction returns YES:

Let φ be a 3-SAT instance and suppose it is satisfiable. We need to show that the system of inequalities we made has a solution. To show this, we should construct x that satisfies the system of inequalities.

Because φ is satisfiable, consider a satisfying assignment. If x_i is assigned true, then in our construction let $x_i = 1$. If x_i is assigned false, let $x_i = 0$.

This assignment clearly satisfies inequalities of the form $x_i \leq 1$ and $x_i \geq 0$. It also satisfies our constraints of the form

$$\sum_{k=1,2,3} \begin{cases} x_{i_k} \text{ if the literal } x_{i_k} \text{ appears in } C_j \\ 1 - x_{i_k} \text{ if the literal } \neg x_{i_k} \text{ appears in } C_j \end{cases} \geq 1$$

because each clause has at least one literal true, which by our construction means at least one of the summands is 1, and the remaining summands are at least 0, so their sum is at least 1.

Problem 3.3 – Write the Proof

d) Show your reduction is correct (⇐). Suppose our reduction returns YES, want to show correct answer is YES:

Problem 3.3 – Write the Proof

d) Show your reduction is correct (⇐). Suppose our reduction returns YES, want to show correct answer is YES:

Suppose that the ILP obtained from converting a 3SAT formula φ returns true, and we need to show that φ is satisfiable. To show this, we need to construct an assignment to the variables of φ.

Because the ILP returned true, there is an integer vector x satisfying our inequalities. If $x_i = 0$, assign x_i to be false. Otherwise, assign x_i to be true.

To show that this is a satisfying assignment, we need to show that it satisfies all clauses. Suppose for contradiction the clause C_j involving variables $x_{i_1}, x_{i_2}, x_{i_3}$ is not satisfied, meaning every literal is false. By our construction, this is because the ILP told us that every positive literal $= 0$ and every negative literal $\neq 0$. Because we had constraints $x_i \geq 0$ and $x_i \leq 1$, this means every negative literal $= 1$. Then,

$$\sum_{k=1,2,3} \begin{cases} x_{i_k} \text{ if the literal } x_{i_k} \text{ appears in } C_j \\ 1 - x_{i_k} \text{ if the literal } \neg x_{i_k} \text{ appears in } C_j \end{cases} = \sum_{k=1,2,3} \begin{cases} 0 \text{ if the literal } x_{i_k} \text{ appears in } C_j \\ 0 \text{ if the literal } \neg x_{i_k} \text{ appears in } C_j \end{cases} = 0$$

which is a contradiction.

That's All, Folks!

Thanks for coming to section this week!
Any questions?