Complexity Examples

Note: All the examples are from chapter 34 of CLRS 3rd edition.

Table of Contents:

Summary of Complexity:

- A **Turing Machine (TM)** solves a dec problem in **polynomial time** if there's a polynomial p s.t. on every instance of n-bit input and m-bit output the TM halts in at most $p(n,m)$ steps.

- A dec problem is **non-deterministic polynomial (NP)** if we can verify an Yes answer in polynomial time.

To prove that a dec problem is in NP, we need to show that there is a polynomial-time algo which:

1. Can accept every Yes instance with the right polynomial-size advice.

2. Will not accept any No instance with any advice.

- A **decision problem** is a problem where the output is Yes/No.

- A dec problem is **CO-NP** if we can verify a No instance in polynomial time.

Note: A dec problem X is in CO-NP iff its complement $\bar{X}$ is NP.

- Problem A is **p-reducible** to Problem B, denoted as $A \leq_p B$ if an oracle/subroutine for B can be used to efficiently solve A.

I.e. You can solve A by making polynomially many calls to an oracle for B and doing addition poly-time computations.

- If $A \leq_p B$ and B can be solved efficiently, then so can A.

- If $A \leq_p B$ and A can't be solved efficiently, then neither can B.

- A problem is NP-hard if we can reduce another NP-hard problem to it.

- A problem is NP-complete if it's both NP and NP-hard.

Question 34.5-2:

Suppose we have a 3-CNF formula F with v literals and c clauses.

Let var $x_i = 1$ if its value is True.
Let var $x_i = 0$ if its value is False.
Let $\overline{x_i}$, the negated version of x_i have value $(1 - x_i)$.

Given this, a clause is True iff the sum of its literals is 1 or more.

E.g. Say we have clause $C_1 = (x_1 \vee x_2 \vee \overline{x_3})$
If any of the 3 literals is True, then C_1 is True.
This also means that $x_1 + x_2 + \overline{x_3} \geq 1$.
If all 3 literals are False, then C_1 is False but also that $x_1 + x_2 + \overline{x_3} = 0$
< 1

$$Ax \geq 1 \Longleftrightarrow (-A)x \leq -1$$

I will create matrix A to be $c \times v$ with each row corresponding to a clause.

Let $a_{ij} = \begin{cases} -1, & \text{if var } j \text{ without negation is in clause } i \\ 1, & \text{if var } \overline{j} \text{ is in clause } i \\ 0, & \text{otherwise} \end{cases}$

Let $\overline{x}$ be a $v \times 1$ vector with each row representing a literal.

Let $\overline{b} = \begin{bmatrix} -1 \\ \vdots \\ -1 \end{bmatrix}$ be a $c \times 1$ vector.

Now, 3-CNF SAT is True iff $Ax \leq b$.

Proof:

(3-CNF SAT $\rightarrow$ 0-1 LP)

Suppose that there's an assignment of literals that make F True.

This means that in each clause, at least one literal is set to True.

Suppose that in clause i, literal $\overline{X_a}$ is True and in Clause j, literal X_b is True, and all other literals in both clauses i and j are False.

With Clause i, we have $a_{ia} = 1$ while $X_a = -1$.
$(a_{ia})(X_a) = -1 \leq -1$

With Clause j, we have $a_{jb} = -1$ while $X_b = 1$.
$(a_{jb})(X_b) = -1 \leq -1$

Since each clause has at least 1 literal whose value is True, $Ax \leq b$.

(0-1 LP $\Rightarrow$ 3-CNF SAT)

Suppose that $Ax \leq b$.
This means that each clause has at least 1 literal who's value is True.
Hence, 3-CNF SAT is True.

Hence, we've proved that 0-1 LP is NP-hard.

Now, I'll prove that 0-1 LP is in NP.
Let the advice be the vector x. We can easily verify whether or not $Ax \leq b$.

Therefore, 0-1 LP is NP-Complete.

Question 34. 5-3:

I'll reduce 0-1 LP to Int Linear Programming.
I.e. 0-1 LP $\leq_p$ Int LP

Given (A, x, b), an instance of 0-1 LP, we want to construct in poly-time (A', x', b') an instance of Int LP, s.t. 0-1 LP is True iff Int LP is True.

Just let $A' = A$, $x' = x$ and $b' = b$.

Note: We also could've done 3-CNF SAT $\leq_p$ Int LP and the soln is the same as before.