Exercise (10.3.2): The problem 4TA-SAT is defined as follows:

Give a prop. formula E, does E have at least 4 satisfying truth assignments?

Show that 4TA-SAT is NP-complete:

Proof:

1) 4TA-SAT is in NP

We devise a non-deterministic poly-time algorithm:

1) Guess 4 truth-assignments T_1, T_2, T_3, T_4
2) Check that T_1, T_2, T_3, T_4 all satisfy E.

Note that both steps require time polynomial in the size of E

2) 4TA-SAT is NP-hard

We show this by reducing SAT to 4TA-SAT.

Let E be a prop. formula, and let $x_1, \dots, x_n$ be all variables in E.

We construct a new formula E' s.t.

$$E \in SAT \iff E' \in 4TA\text{-}SAT$$

Let y_1, y_2 be two new variables. Then

$$E' = E \vee ((x_1 \wedge x_2 \wedge \dots \wedge x_n) \wedge ((y_1 \wedge y_2) \vee (y_1 \wedge \overline{y_2}) \vee (\overline{y_1} \wedge y_2)))$$

Consider the truth assignments for $x_1, \dots, x_n, y_1, y_2$

	$x_1 \dots x_i \dots x_n$	y_1	y_2	Case 1: $E \notin SAT$ E	Case 1: $E \notin SAT$ E'	Case 2: $E \in SAT$ E	Case 2: $E \in SAT$ E'
1)	T ... T	T	T	F	T	?	T
2)	T ... T	T	F	F	T	?	T
3)	T ... T	F	T	F	T	?	T
4)	T ... T	F	F	F	F	⋮	⋮
	... F ...		...	F	F		
	⋮			⋮	⋮	E	T
2^{n+2}				F	F	⋮	⋮
					≥ 3		≥ 4

Alternative solution:

$$E' = E \wedge (y_1 \vee y_2 \vee y_3)$$

- If E is unsatisfiable, then E' is unsatisfiable, and hence $E' \notin 4TA\text{-}SAT$
- If E is satisfiable, then E' has at least 7 satisfying truth assignments: these are obtained by combining
  - a TA for $x_1, \dots, x_n$ satisfying E with
  - the 7 TAs for y_1, y_2, y_3 satisfying $y_1 \vee y_2 \vee y_3$

Exercise 11.9.9 b)

Consider the problem FALSE-SAT:

Given a boolean expression E that is false when all its variables are made false, is there some other truth assignment that makes E false, besides all-false?

Decide whether the problem is in NP or coNP.

Describe its complement.

If the problem or its complement is NP-complete, prove it.

Proof:

The problem is NP-complete.

- In NP: given a boolean expression E, we need to check:
  1) that E is false when all variables are assigned false
  2) that there is some other truth-assignment making E false

  (1) can be done in poly-time by a DTM
  (2) can be done in poly-time by a NTM
  guess a truth-assignment T different from all false, and answer yes if under T, E evaluates to false

- NP-hard: by a reduction from SAT

  Let E be a boolean expression with variables $x_1, \dots, x_n$

  We construct an expression E' s.t. $E \in SAT$ iff $E' \in$ FALSE-SAT

  1) Test if E is true when all variables are false (polynomial)
  If so, $E \in SAT$, and we convert it to a fixed expression that is in FALSE-SAT, e.g. $x \wedge y$.

2) Otherwise, let E' be $\neg E \wedge (x_1 \vee x_2 \vee \cdots \vee x_n)$.

Clearly, the reduction is poly-time.

We have that E' is false when all of $x_1, \ldots, x_n$ are false.

Notice that in case (2), E is false when all variables are false.

Hence, if $E \in SAT$, then it is satisfied by a truth assignment T different from all-false.

Thus, $\neg E$ is made false by T, and $E' \in FALSE\text{-}SAT$.

Conversely, if $E' \in FALSE\text{-}SAT$, then since $x_1 \vee \cdots \vee x_n$ is false only for the all-false truth assignment, there must be some other truth-assignment T that makes $\neg E$ false. Then T makes E true, and $E \in SAT$.

Exercises on problems in P, NP, and NP-complete

Exercise 1:

Consider the following optimization version of SAT:

MAXSAT: Input: a propositional formula F in CNF, and an integer k

Output: yes, if there is a truth assignment that satisfies at least k clauses of F

no, otherwise

What is the complexity of MAXSAT?

a) MAXSAT $\in$ NP: immediate, by the following NP algorithm

1) guess a truth assignment α (non-deterministic polynomial)
2) count the # of clauses satisfied by α, and answer yes iff it is $\geq k$ (deterministic polynomial)

b) MAXSAT is NP-hard

This follows from the fact that CSAT is a special case of MAXSAT.

Formally, we can polynomially reduce CSAT to MAXSAT, i.e.

$$SAT <_{poly} MAXSAT$$

Given an instance F of CSAT, we construct an instance (F, k) of MAXSAT, where k is the # of clauses of F. Obviously, k can be obtained in poly time from F, and

$$F \in CSAT \iff (F, k) \in MAXSAT$$

□

Exercise 2:

Show that CSAT remains NP-complete even if it is restricted to instances in which each variable appears at most three times. Let's call this variant 3VARCSAT

Solution: We show how to reduce CSAT to 3VARCSAT.

Given a formula F in CNF, we construct a formula E in CNF where each variable appears at most three times and such that E is satisfiable iff F is satisfiable.

Let x be a variable appearing in F k times, with $k > 3$. Then we construct from F a formula F_x as follows:

1) We replace the i-th occurrence of x in F with x_i, for $i \in \{1, \dots, k\}$, where each x_i is a fresh variable.
2) We add the following clauses, ensuring that all variables $x_1, \dots, x_k$ are assigned the same truth value

$$(\overline{x}_1 \vee x_2) \wedge (\overline{x}_2 \vee x_3) \wedge \dots \wedge (\overline{x}_{k-1} \vee x_k) \wedge (\overline{x}_k \vee x_1)$$

We have that F_x can be constructed from F in polynomial time, and F_x is satisfiable iff F is satisfiable.

Then, the formula E is obtained from F by repeating the above transformation for each variable x occurring in F more than three times.

We have that:

1) Each variable appears in E at most three times
2) E can be constructed from F in polynomial time
3) E is satisfiable iff F is satisfiable

□

Exercise 3

Consider 2VARCSAT, i.e. the variant of CSAT in which each variable appears at most two times.

What is the complexity of 2VARCSAT.?

Solution: 2VARCSAT is in P.

Let F be an instance of 2VARCSAT.

Notice that for each variable x, we have one of 3 cases:

1) x appears only positively in F (one or two times)
2) x " " negatively in F (" ")
3) x appears one time positively and one time negatively in F

From this, we obtain the following algorithm to decide the satisfiability of F:

Input: set F of clauses over variables $x_1, \dots, x_n$, with each x_i appearing in at most two clauses

Output: YES, if F is satisfiable, NO otherwise

For each variable $x \in \{x_1, \dots, x_n\}$

if x appears only positively in F (case 1), or
x appears only negatively in F (case 2), or
x appears only in one clause of F
then remove from F the clause(s) in which x_i appears
else let $C = x \vee C_{rest}$ and $C' = \bar{x} \vee C'_{rest}$ (case 3)
be the two clauses of F in which x appears
if C_{rest} and C'_{rest} are both empty
then answer NO
else remove from F both C and C', and
replace them with the clause $C_{rest} \vee C'_{rest}$

Answer YES

The algorithm runs in polynomial time, since the for-loop is executed n times, and each iteration is at most linear.

Note that a variable x_i might be removed from F before it is considered in the i-th iteration of the for loop. In this case, F is not changed in that iteration.

Moreover:

- For cases (1) and (2) the clauses containing x can be trivially satisfied by making x true/false.
- For case (3), the algorithm applies a resolution step to x, and replaces the clauses C and C' with their resolvant.
- By applying a resolution step to a variable x_i, for another variable x_j that has not yet been considered (i.e. $j \geq i$) and that occurred positively and negatively in two different clauses, the two occurrences of x_j might be merged into a single clause $C_{rest} \vee C'_{rest}$.
  This clause can be removed, since it is trivially satisfied by every truth assignment.