

Exercises on problems in P, NP, and NP-completeExercise 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 (nondeterministic 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.

$$\text{SAT} \leq_{\text{poly}} \text{MAXSAT}$$

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 polytime from F , and

$$F \in \text{CSAT} \iff (F, k) \in \text{MAXSAT}$$

□

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 $k \geq 3$.

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 appears only negatively in F (one or two times)

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 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 resolvent.
- By applying a resolution step to a variable x_i , for another variable x_j that has not yet been considered (i.e. $j > i$) and that occurred positively and negatively in two different clauses, the two occurrences of x_j might be merged into a single clause $C_{\text{res}} \vee C'_{\text{res}}$.

This clause can be removed, since it is trivially satisfied by every truth assignment.

Consider the following problems:

1) Vertex-cover (VC)

Given an undirected graph $G=(V,E)$ and an integer $k \geq 2$, is there a subset C of V with $|C| \leq k$ such that C covers all edges of G (i.e., for each edge $\{v_i, v_j\} \in E$ with $v_i \neq v_j$, $\{v_i, v_j\} \cap C \neq \emptyset$).

2) Independent-set (IS)

Given an undirected graph $G=(V,E)$ and an integer $k \geq 2$, is there a subset C of V with $|C| \geq k$ such that for all $v_i, v_j \in C$ with $v_i \neq v_j$, $\{v_i, v_j\} \notin E$.

3) Clique

Given an undirected graph $G=(V,E)$ and an integer $k \geq 2$, is there a subset C of V with $|C| \geq k$ such that for all $v_i, v_j \in C$ with $v_i \neq v_j$, $\{v_i, v_j\} \in E$.

Show that VC, IS, and Clique can be reduced to each other in polynomial time.

N.B. In the definitions of VC, IS, and Clique we have ignored self-loops (since we required $v_i \neq v_j$).

IS \leq_{poly} Clique

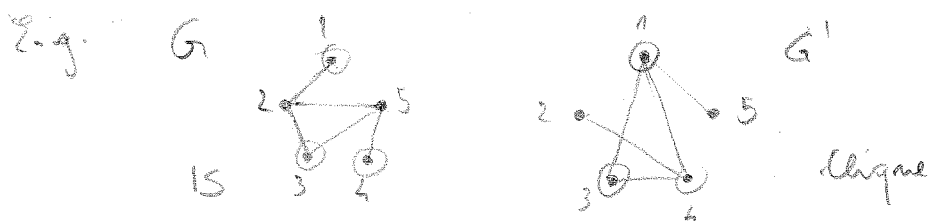
Given an instance (G, k) of IS, we construct an instance (G', k') of Clique as follows:

$$k' = k$$

Let $G = (V, E)$.

Then $G' = (V, E')$, where $E' = V \times V \setminus E$

(i.e., the edges of E' are obtained by connecting all pairs of nodes that are not connected in E .)



The reduction works because the maximum independent set of G is precisely the maximum clique in the complement graph of G .

VC \leq_{poly} IS

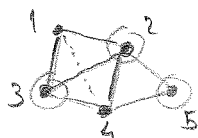
Given an instance (G, k) of VC, we construct an instance (G', k') of IS as follows:

$$k' = |V| - k$$

$$G' = G$$

The reduction works, because the vertices in a vertex-cover C cover all edges of G . Hence the set $V \setminus C$ must have no edges between its elements, and is thus an independent set.

E.g.



The marked nodes $\{2, 3, 5\}$ are a VC of size 3. Hence, there cannot be an edge $\{1, 4\}$

Clique $\xrightarrow{\text{poly}}$ VC

Given an instance (G, k) of Clique, we construct an instance (G', k') of VC as follows:

$$k' = |V| - k$$

Let $G = (V, E)$.

Then $G' = (V, E')$, where $E' = V \times V \setminus E$.