Computational Complexity

Exercise Session 2

Note: these solutions are (often) merely pointers to the right idea that is needed to solve the problems. These are not fully worked-out solutions. So please do not take these solutions as an example for how to write up your solutions for, e.g., the homework assignments. :-)

Exercise 1. Show that coNP $\subseteq$ EXP.

Solutions:

See the proof of Claim 2.4 in Arora & Barak (2009).

Exercise 2. Consider the following problem Reverse-3SAT:

> **Instance:** A propositional formula φ in 3CNF—that is, a formula of the form $\varphi = c_1 \wedge \cdots \wedge c_m$, where each c_j is of the form $c_j = \ell_{j,1} \vee \ell_{j,2} \vee \ell_{j,3}$, where $\ell_{j,1}, \ell_{j,2}, \ell_{j,3}$ are propositional literals.
>
> **Question:** Is there a truth assignment α to the variables occurring in φ that sets at least one literal in each clause c_j to **false**?

Prove that Reverse-3SAT is NP-complete—that is, prove that is in NP and that it is NP-hard. To show NP-hardness, you may give a reduction from any known NP-complete problem.

- *Hint:* reduce from 3SAT.

Solutions:

To show NP-hardness, reduce from 3SAT and simply negate each literal occurrence in each clause.

Exercise 3. Consider the following problem CLIQUE:

> **Instance:** An undirected graph $G = (V, E)$, and a positive integer $k \in \mathbb{N}$.
>
> **Question:** Does G contain a clique of size k—that is, is there a set $C \subseteq V$ of vertices with $|C| = k$ such that for each $v, v' \in C$ with $v \neq v'$ it holds that $\{v, v'\} \in E$?

In this exercise, we will show that CLIQUE is NP-complete.

(i) Prove that CLIQUE is in NP.

Solutions:

To show NP-hardness, use as certificate a binary string that indicates the set $C \subseteq V$, and check that it is of the right size and that it is a clique (both polynomial-time computable).

To show that CLIQUE is NP-hard, we will give a polynomial-time reduction f from 3SAT to CLIQUE. We describe this reduction f as follows: for an arbitrary instance φ of 3SAT, we describe what the instance $f(\varphi) = (G, k)$ looks like.

Let $\varphi = c_1 \wedge \cdots \wedge c_m$ be an arbitrary 3CNF formula, containing propositional variables $x_1, \ldots, x_n$, where $c_j = \ell_{j,1} \vee \ell_{j,2} \vee \ell_{j,3}$ for each $1 \leq j \leq m$. Then we construct the graph $f(\varphi)$ as follows.

- We introduce vertices $v_{j,1}, v_{j,2}, v_{j,3}$, for each $1 \leq j \leq m$. That is, for each clause c_j we add three vertices—one for each literal occurring in the clause.
- Two vertices $v_{j,l}$ and $v_{j',l'}$ are connected with an edge if and only if $j \neq j'$ and the literals $\ell_{j,l}$ and $\ell_{j',l'}$ are not each other's negation.

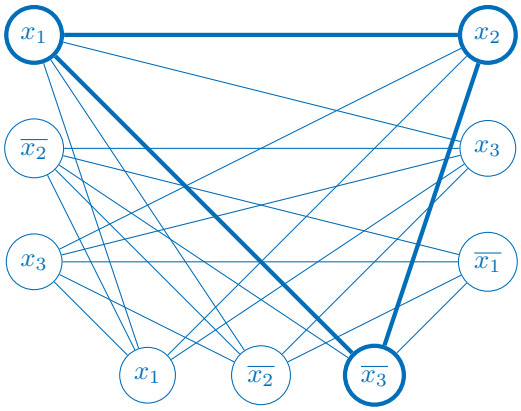
Finally, we set $k = m$.

Let $\varphi_{\text{ex}} = (x_1 \vee \overline{x_2} \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_2 \vee x_3 \vee \overline{x_1})$ be an example 3CNF formula.

(ii) Let $f(\varphi_{\text{ex}}) = (G_{\text{ex}}, k_{\text{ex}})$. Compute k_{ex} and draw the graph G_{ex}.

Solutions:

$k_{\text{ex}} = 3$ and G_{ex} looks as follows (together with a 3-clique).



(iii) Show that φ_{ex} is satisfiable. Use a satisfying assignment for φ_{ex} to produce a clique of size k_{ex} for G_{ex}.

Solutions:

For example, take the truth assignment $\alpha = \{x_1 \mapsto 1, x_2 \mapsto 1, x_3 \mapsto 0\}$, corresponding to the 3-clique drawn before.

(iv) Prove, for an arbitrary 3CNF formula φ, that φ is satisfiable if and only if $f(\varphi) = (G, k) \in \mathsf{CLIQUE}$.

Solutions:

Since nodes corresponding to literals in the same clause are not connected to each other, finding a clique of size m requires picking one vertex for each clause (picking a literal in the clause to satisfy). Then, since contradictory literals are not connected to each other by an edge, any m-clique corresponds to a set of collectively satisfiable literals, one for each clause. In other words, m-cliques correspond to satisfying (partial) truth assignments.

(v) Explain why the function f is polynomial-time computable.

Solutions:

You need $3m$ vertices, and for each pair of vertices you can check whether there should be an edge between them or not by looking at what literals they correspond to (which can be done in polynomial time).

Exercise 4 (reduction from HamCycle to HamPath)**.** Consider the following problem HamPath:

> **Instance:** An undirected graph $G = (V, E)$, and two vertices $s, t \in V$.
>
> **Question:** Is there a Hamiltonian path in G from s to t—in other words, a path from s to t that visits each vertex exactly once?

Consider also the following problem HamCycle:

Instance: An undirected graph $G = (V, E)$.

Question: Is there a Hamiltonian cycle in G—in other words, a cycle that visits each vertex exactly once?

Give a polynomial-time reduction from HamCycle to HamPath.

Solutions:

Take an instance of HamCycle, consisting of a graph $G = (V, E)$. We will construct a graph $G' = (V', E')$, as follows. Pick some arbitrary vertex $v \in V$. Add three fresh vertices v', s, t to V, forming V'. We let E' consist of E together with the following edges. Whenever $\{v, u\} \in V$, we add $\{v', u\}$ to E'. Moreover, we add $\{s, v\}$ and $\{t, v'\}$.

The claim is then that there is a Hamiltonian path from s to t in G' if and only if G has a Hamiltonian cycle. (This needs to be proven, but is straightforward.)

Exercise 5 (self-reducibility of 3SAT)**.** Suppose that you have a polynomial-time algorithm A for (the decision problem) 3SAT. Show that you can use A to construct a polynomial-time algorithm B that, when given as input a 3CNF formula φ, outputs a satisfying assignment α for φ if such an assignment exists, and that outputs 0 otherwise.

Solutions:

See the proof of Theorem 2.18 in Arora & Barak (2009).