More NP–Completeness Proofs

Preliminaries

The reduction technique revisited: a way to show that a problem is **NP**-complete is to use the deduction rule below.

$$\frac{\begin{array}{c} L \text{ is } \mathbf{NP}\text{-complete} \\ L' \text{ is in } \mathbf{NP} \\ \text{There is a mapping } f \text{ such that } x \in L \text{ iff } f(x) \in L' \text{ holds for all } x \\ f(x) \text{ can be be computed in logarithmic space w.r.t. } |x| \end{array}}{L' \text{ is } \mathbf{NP}\text{-complete}}$$

A Proof that FEEDBACK VERTEX SET is NP-complete

Preliminaries

The problem NODE COVER is the following:

> Given an undirected graph $G = \langle V, E \rangle$ and an integer $B \leq |V|$, is there a subset $V' \subseteq V$, $|V'| \leq B$, such that each edge in G has at least one of its endpoints in V'.

We know that NODE COVER is **NP**-complete (see page 190 in the book).

The Problem and a Solution

Show that the following problem, called FEEDBACK VERTEX SET, is **NP**-complete:

> Given a directed graph $G = \langle V, E \rangle$ and an integer $B \leq |V|$, is there a subset $V' \subseteq V$ such that (i) $|V'| \leq B$, and (ii) every directed circuit in G includes at least one vertex from V'?

Note that an equivalent definition of the problem is: Given a directed graph $G = \langle V, E \rangle$ and an integer $B \leq |V|$, is there a subset $V' \subseteq V$ such that (i) $|V'| \leq B$, and (ii) the directed graph $G' = \langle V \setminus V', E \cap (V \setminus V') \times (V \setminus V') \rangle$ has no cycles?

Obviously FEEDBACK VERTEX SET is in **NP**: (i) guess the set V' such that $|V'| \leq B$, and (ii) use, e.g., Tarjan's algorithm for finding strongly connected components of a graph to check whether $G' = \langle V \setminus V', E \cap (V \setminus V') \times (V \setminus V') \rangle$ has any cycles.

We show the **NP**-hardness by reducing from the **NP**-complete problem NODE COVER. Given an input $G; B$ for NODE COVER, where $G = \langle V, E \rangle$ is an undirected graph and $B \leq |V|$ is an integer, the corresponding input for FEEDBACK VERTEX SET is $\hat{G}; B$, where $\hat{G}$ is the directed graph $\hat{G} = \langle V, \hat{E} = \{\langle v_i, v_j \rangle, \langle v_j, v_i \rangle \mid \{v_i, v_j\} \in E\} \rangle$. That is, $\hat{G}$ is G interpreted as a directed graph.

Now if G has a node cover V', $|V'| \leq B$, then the directed graph $G' = \langle V \setminus V', \hat{E} \cap (V \setminus V') \times (V \setminus V') \rangle$ has no edges and thus cannot have any cycles. Thus if V' is a node cover for G, then V' is a feedback vertex set for $\hat{G}$.

On the other hand, assume that $\hat{G}$ has a feedback vertex set V', $|V'| \leq B$. Then for each edge pair $\langle v_i, v_j \rangle$ and $\langle v_j, v_i \rangle$ between the nodes v_i and v_j, at least one of v_i, v_j has to be in V' (otherwise $G' = \langle V \setminus V', \hat{E} \cap (V \setminus V') \times (V \setminus V') \rangle$ would have a cycle $v_i \to v_j \to v_i$). Therefore V' is also a node cover for G.

To sum up, V' is a node cover of G iff V' is a feedback vertex set of $\hat{G}$. Obviously the reduction can be computed in logarithmic space.

A Proof that PARTITION INTO TRIANGLES is NP-complete

Preliminaries

Graphs in this exercise are assumed to be undirected and not containing self-loops.

The problem EXACT COVER BY 3-SETS is the following:

> We are given a family $F = \{S_1, \ldots, S_n\}$ of subsets of a set U, such that $|U| = 3m$ for some integer m, and $|S_i| = 3$ for all i. The question is whether there are m sets in F that are disjoint and have U as their union.

We know that EXACT COVER BY 3-SETS is **NP**-complete (see page 201 in the book).

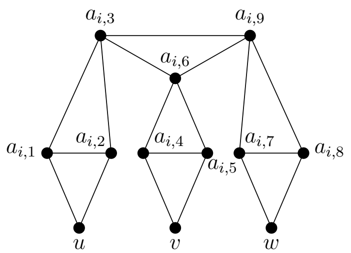
The Problem and a Solution

Show that the following problem, called PARTITION INTO TRIANGLES, is **NP**-complete:

> Given a graph $G = \langle V, E \rangle$ with $|V| = 3q$ for an integer q, is there a partition of V into q mutually disjoint sets $V_1, V_2, \ldots, V_q$ of three vertices each such that, for each $V_i = \{v_{i,1}, v_{i,2}, v_{i,3}\}$, the three edges $\{v_{i,1}, v_{i,2}\}$,$\{v_{i,1}, v_{i,3}\}$, and $\{v_{i,2}, v_{i,3}\}$ all belong to E?

Clearly PARTITION INTO TRIANGLES is in **NP** because we can first non-deterministically guess a partition of V and then check (in deterministic polynomial time) that each V_i fulfills the triangle condition.

We show the **NP**-hardness by reducing from the **NP**-complete problem EXACT COVER BY 3-SETS. Assume that we are given a set U such that $|U| = 3m$ for some integer m and a family $F = \{S_1, \ldots, S_n\}$ of subsets of U such that $|S_i| = 3$ for all i. We now construct a graph $G = \langle V, E \rangle$ with $|V| = 3q'$ such that G can be partitioned into q' triangles iff F contains an exact cover for U. First, it can be safely assumed that each element $u \in U$ appears in at least one set S_i in F: if this is not the case, then F cannot have an exact cover for U and we can simply output a simple constant graph that cannot be partitioned into triangles. Otherwise, we substitute for each set $S_i = \{u, v, w\}$ appearing in F the collection E_i of 18 edges shown below.



The graph $G = \langle V, E \rangle$ is now defined by

$$\begin{aligned} V &= U \cup \bigcup_{i=1}^{n} \{a_{i,j} \mid 1 \leq j \leq 9\} \\ E &= \bigcup_{i=1}^{n} E_i. \end{aligned}$$

Now $|V| = |U| + 9n = 3m + 9n$ and thus $q' = m + 3n$. Note that only the U-vertices may be shared by different E_is.

Assume that $F' \subseteq F$ is an exact partition of U (and thus $|F'| = m$). Then, if $S_i = \{u, v, w\} \in F'$, take $\{a_{i,3}, a_{i,6}, a_{i,9}\}$, $\{u, a_{i,1}, a_{i,2}\}$, $\{v, a_{i,4}, a_{i,5}\}$ and $\{w, a_{i,7}, a_{i,8}\}$ to belong to the triangle partition of G and, if $S_i = \{u, v, w\} \notin F'$, choose $\{a_{i,1}, a_{i,2}, a_{i,3}\}$, $\{a_{i,4}, a_{i,5}, a_{i,6}\}$ and $\{a_{i,7}, a_{i,8}, a_{i,9}\}$ to belong to the triangle partition of G. Clearly the result is a valid triangle partition of G because the partition is disjoint, each vertex is in the partition and the partition has $q' = m + 3n$ sets.

To show the opposite direction, assume that V can be partitioned into $q' = m+3n$ triangles $V_1, V_2, \ldots, V_{q'}$. Then the corresponding exact cover of U is given by choosing those S_i of F such that $\{a_{i,3}, a_{i,6}, a_{i,9}\} = V_k$ for some $1 \leq k \leq q'$. This is justified as follows. For each $u \in U$ there is exactly one triangle into which u belongs, let's say it was $\{u, a_{i,1}, a_{i,2}\}$. Then $\{a_{i,3}, a_{i,6}, a_{i,9}\}$, $\{v, a_{i,5}, a_{i,6}\}$ and $\{w, a_{i,6}, a_{i,9}\}$ for $S_i = \{u, v, w\}$ must also be in the triangle partition. Furthermore, since no other $\{u, a_{j,x}, a_{j,y}\}$ can be in the triangle partition, the corresponding $\{a_{j,3}, a_{j,6}, a_{j,9}\}$ cannot be in the triangle partition. Therefore each u appears in exactly one chosen S_i. This also implies that we choose exacly m S_is.

We have thus showed that G can be partitioned into $q' = m + 3n$ triangles iff F contains an exact cover of m sets for U.

To show that the reduction can be carried out in logarithmic space, note that the reduction is made by local substitutions. We therefore need 3 registers of length $\log(3m)$ to remember the elements in the currently processed set S_i of F and one counter of length $\log n$ to remember the number i of the set. We then just output the edges in E_i to the output tape.

Two Exercises from Chapter 10

10.4.4

Preliminaries

First notice that in resolution we see a clause as a set of literals. Now assume a set ϕ of clauses (a formula in CNF). Consider two clauses $C = \{x\} \cup C'$ and $D = \{\neg x\} \cup D'$ in ϕ, where C' and D' contain the rest of the literals in the clauses C and D, respectively. The *resolution rule* says that one can infer $C' \cup D'$ from C and D. One application of the resolution rule is called a *resolution step*, and the clause $C' \cup D'$ is called a *resolvent* of C and D.

We know that a set of clauses ϕ is unsatisfiable iff it has a *resolution refutation*, i.e., a sequence of clauses $(C_0, \ldots, C_n)$ such that

(a) every C_i is either

 (i) a clause in ϕ, or

(ii) a resolvent of some two clauses C_j, C_l, where $j, l \in \{0, \ldots, i-1\}$,

and

(b) $C_n = \emptyset$.

The length of a resolution refutation is n, i.e., the number of clauses in it.

The *polynomial resolution conjecture* claims that any unsatisfiable CNF ϕ has a resolution refutation of *polynomial* length in the size of ϕ.

The problem

Show that the polynomial resolution conjecture implies $\mathbf{NP} = \mathbf{coNP}$.

A solution

Take an expression (a set of clauses) ϕ of size n. Then, in case ϕ is unsatisfiable, ϕ has a resolution refutation $(C_0, \ldots, C_{n^k})$ for some fixed k. Construct a non-deterministic polynomial time Turing machine M that, given ϕ as input, (i) first guesses $(C_0, \ldots, C_{n^k})$, (ii) then (deterministically) checks that $(C_0, \ldots, C_{n^k})$ is a valid resolution refutation for ϕ (this can be done in polynomial time) and (iii) finally says "yes" if this was the case ("no" otherwise). Clearly M says "yes" iff ϕ is unsatisfiable. Therefore the complement of SAT, $\overline{\text{SAT}}$, is in $\mathbf{NP}$.

The fact that $\overline{\text{SAT}} \in \mathbf{NP}$ means that $\mathbf{coNP} \subseteq \mathbf{NP}$ since $\overline{\text{SAT}}$ is complete for $\mathbf{coNP}$ (and both $\mathbf{NP}$ and $\mathbf{coNP}$ are closed under reductions).

Now take any language $L \in \mathbf{NP}$. We can reduce L to SAT since SAT is $\mathbf{NP}$-complete. Since $\overline{\text{SAT}} \in \mathbf{NP}$, SAT $\in \mathbf{coNP}$ by the definition of complement classes and therefore $L \in \mathbf{coNP}$. Thus $\mathbf{NP} \subseteq \mathbf{coNP}$.

To sum up, if the polynomial resolution conjecture were true, $\mathbf{NP} = \mathbf{coNP}$.

A note: The polynomial resolution conjecture was disproved by Haken in Theoretical Computer Science 39:297–308, 1985.

10.4.6

Preliminaries

A *strong* non-deterministic Turing machine (SNTM)

- has three possible outcomes (halting states): "yes", "no" and "maybe"

- decides a language L if the following holds:
  - If $x \in L$, then all computations end up with "yes" or "maybe", and at least one with "yes".
  - If $x \notin L$, then all computations end up with "no" or "maybe", and at least one with "no".

The problem

Show that L is decided by a strong non-deterministic machine in polynomial time if and only if $L \in \mathbf{NP} \cap \mathbf{coNP}$.

A solution

(The "only if" direction) Let L be a language that is decided in polynomial time by an SNTM M. Now L belongs to $\mathbf{NP}$ because we can build a normal non-deterministic Turing machine (NTM) $M_{\mathbf{NP}}$ by replacing each "maybe" halting state by "no". Now $M_{\mathbf{NP}}$ clearly decides L polynomial time i.e. $L \in \mathbf{NP}$. Similarly, L belongs to $\mathbf{coNP}$ since we can build a normal NTM $M_{\mathbf{coNP}}$ by (i) swapping the halting states "no" and "yes" and (ii) replacing each "maybe" halting state by "no". Now clearly $M_{\mathbf{coNP}}$ decides the complement of L in polynomial time meaning that $L \in \mathbf{coNP}$. Therefore if L is decided by an SNTM, then L belongs to $\mathbf{NP} \cap \mathbf{coNP}$.

(The "if" direction) Let $L \in \mathbf{NP} \cap \mathbf{coNP}$. Let $M_{\mathbf{NP}}$ be a normal NTM that decides L in polynomial time and $M_{\mathbf{coNP}}$ a normal NTM that decides the complement of L in polynomial time. Construct M that is the "cross product" of $M_{\mathbf{NP}}$ and $M_{\mathbf{coNP}}$, executing both machines synchronously in parallel. Define the halting states of M in the following way: (i) if the $M_{\mathbf{NP}}$-part of M says "yes", then M says "yes" (ii) if the $M_{\mathbf{coNP}}$-part of M says "yes", then M says "no" and (ii) both parts say "no", then M says "maybe". Note that items (i), (ii) and (iii) are disjoint when a single computation path of M is considered. M clearly decides L since if $x \in L$, then there is at least one path in M in which the $M_{\mathbf{NP}}$ part says "yes" and therefore M also says "yes"; and furthermore, M cannot say "no" because $M_{\mathbf{coNP}}$ cannot say "yes" for $x \in L$. Symmetric argumentation holds for the case $x \notin L$. Therefore, if $L \in \mathbf{NP} \cap \mathbf{coNP}$, then there is an SNTM that decides L.

As conclusion, L is decided by a SNTM iff $L \in \mathbf{NP} \cap \mathbf{coNP}$.