Computability and Complexity

Instituto de Matemáticas

Exercises

Time complexity

1 P

Exercise 1 : *Connections*.
Prove that $L_C = \{< G > \mid G \text{ is a connected undirected graph}\}$ is in P.
Answer:
We devise the following algorithm:

1. *Check that G is an undirected graph.*
2. *Mark the first vertex.*
3. *Repeat until no more vertex is marked:*
   (a) *Scan the edges. For every edge uv such that u is marked and v is unmarked, mark v.*
4. *If all vertices are marked, accept; otherwise, reject.*

Step 1 is polynomial ($O(n^2)$), step 2 is constant. Step 3 can happen at most n times, and does a constant operation for every edge, thus step 3 is polynomial ($O(n^3)$). Finally, step 4 is polynomial ($O(n)$).
Furthermore, it is easy to see that the algorithm verifies if a graph is connected: if G is connected, then all vertices will end up marked; and if G is not connected, then some vertex will be unmarked. Thus, L_C is in P.

□

Exercise 2 : *Isoceles*.
Prove that $L_\Delta = \{< G > \mid G \text{ is an undirected graph containing a triangle}\}$ is in P.
Answer:
The idea is to test all possible triangles, that is, all sets of three vertices. We have the following algorithm:

1. *Check that G is an undirected graph with at least 3 vertices.*
2. *For all sets of three distinct vertices $\{u, v, w\}$:*
   (a) *Scan the edges of G. If uv, vw and uw are found, accept.*
3. *Reject.*

It is easy to see that this algorithm will accept a graph containing a triangle, and reject a triangle-free graph. Now, let us study its complexity. Step 1 is polynomial ($O(n^2)$) and step 3 is constant. Step 2 can be executed, in the worst case, for every set of 3 vertices, so $\binom{n}{3} = \frac{(n-2)(n-1)n}{6} = O(n^3)$ times. Each execution is polynomial ($O(n^2)$), so step 2 is polynomial ($O(n^5)$). Thus, L_Δ is in P.

□

Exercise 3 : *Modular exponentiation*.
Prove that $L_{\text{mod}} = \{< b, e, c, p > \mid b, e, c, p \text{ are binary integers and } b^e \equiv c \bmod p\}$ is in P.
Answer:
The idea here is to use the fact that $(x \times y) \bmod m = ((x \bmod m) \times (y \bmod m)) \bmod m$. Furthermore, since the numbers are in binary, we have $b^e = b^{\sum_{i=0}^{n} e_i 2^i} = \prod_{i=0}^{n} (b^{2^i})^{e_i}$.
We have the following algorithm:

1. *Check that there are four binary integers and that $p \neq 0$. If $p = 1$, check whether $c = 0$.*
2. *Set $r := 1$ and $b := b \bmod p$.*
3. *For i that goes from 0 to n, do the following:*
   (a) *If $e_i = 1$, then set $r := (r \times b) \bmod p$.*
   (b) *Set $b := (b \times b) \bmod p$.*
4. *Check whether $c = r$. If so, accept; otherwise, reject.*

Remember that the multiplication and modulo operations can be done in polynomial time. Thus, the algorithm has n executions of one multiplication and two modulo operations, where n is the size of e in binary. This implies that the algorithm is polynomial. It is easy to check that it computes the modular exponentiation $b^e \bmod p$, and then check whether the result is c or not. Hence, L_{mod} is in P.

□

Exercise 4 : *Unary is more efficient than binary?!*.
In the class, we saw that SUBSET-SUM is NP-complete. However, consider UNARY-SUBSET-SUM, an instance of SUBSET-SUM where the numbers are written in unary. Prove that UNARY-SUBSET-SUM is in P.
Answer:
The idea is to convert the input in binary, then simulate a nondeterministic Turing machine that solves SUBSET-SUM on the converted input. The algorithm is as follows:

1. *Convert the input in binary.*
2. *Simulate a nondeterministic Turing machine that solves SUBSET-SUM.*
3. *Return the same output that the simulation.*

Step 1 is polynomial, and convert the input which is of size n to a $O(log(n))$ intermediary input for the simulation. Now, step 2 is polynomial in the nondeterministic Turing machine (since SUBSET-SUM is in NP), so it is nondeterministically done in $O(log(n)^k)$. As we saw in the course, there is a deterministic Turing machine that simulates our nondeterministic machine with an exponential factor, so step 2 can be done in $O(2^{log(n)^k}) = O(n^k)$, that is, step 2 is polynomial. Finally, step 3 is constant. Thus, UNARY-SUBSET-SUM is in P.

□

2 NP-completeness

Exercise 5 : *Double-SAT*.
Prove that D-SAT= $\{< \phi > \mid \phi$ is a boolean formula with at least two satisfying assignments$\}$ is NP-complete.
Answer:
D-SAT is clearly in NP: a certificate is two assignments of the variables, so a verifier will simply check that the two assignments are distinct and that they satisfy ϕ, which is polynomial (alternatively, we can decide the language by nondeterministically testing all possible pairs of distinct assignments and accepting if a pair satisfies ϕ, which can be done in polynomial time).
We then do a reduction from SAT. Let $\phi(x_1, \ldots, x_n)$ be a boolean formula over variables $x_1, \ldots, x_n$. We define f the following way: let y be a new variable, let $f(\phi(x_1, \ldots, x_n)) = \phi(x_1, \ldots, x_n) \wedge (y \vee \overline{y})$. It is easy to verify that f is a polynomial reduction. Assume that ϕ has a satisfying assignment, then $f(\phi)$ has two satisfying assignments: for both, keep the same values for the x_i's, and then in one assignment we set y as `true` and in the second we set y as `false`. Conversely, if ϕ does not have a satisfying assignment, then $f(\phi)$ clearly will not have a satisfying assignment either.
The above proves that SAT $\leq_p$ D-SAT, and since SAT is NP-complete and D-SAT is in NP, this proves that D-SAT is NP-complete.

□

Exercise 6 : *Partition*.
Prove that PARTITION= $\{< A > \mid A$ is a set of integers such that there exists $A' \subseteq A$ s.t. $\sum_{a\in A'} a = \sum_{a\in A\setminus A'} a\}$ is NP-complete.
Answer:
PARTITION is clearly in NP: a certificate is a subset A' so a verifier will check that $\sum_{a\in A'} a = \sum_{a\in A\setminus A'} a$, which can be done in polynomial time.
We then do a reduction from SUBSET-SUM. Let (A, x) be an input for SUBSET-SUM, and let $s = \sum_{a\in A} a$. We define the following reduction: $f(A, x) = B = A \cup \{s - 2x\}$. The reduction is polynomial, and now we prove that it is correct.
First, assume that there is $A' \subseteq A$ such that $\sum_{a\in A'} a = x$. Then, we set $B' = A' \cup \{s - 2x\}$ and we have $\sum_{a\in B'} a = \sum_{a\in A'} a + s - 2x = s - x = \sum_{a\in A} a - \sum_{a\in A'} a = \sum_{a\in B\setminus B'} a$.
Conversely, assume that there is $B' \subseteq B$ such that $\sum_{a\in B'} a = \sum_{a\in B\setminus B'} a$. Without loss of generality, we can assume that $s - 2x \in B'$, and thus this partition implies that there is $A' \subseteq A$ such that $s - 2x + \sum_{a\in A'} a = \sum_{a\in A\setminus A'} a$. This, in turn, implies that $s - 2x + 2\sum_{a\in A'} a = \sum_{a\in A} a = s$ and thus $\sum_{a\in A'} a = x$.
The reduction being correct, we have SUBSET-SUM $\leq_p$ PARTITION and PARTITION is in NP, which implies that PARTITION is NP-complete.

□

Exercise 7 : *Half-clique*.
Prove that HALF-CLIQUE= $\{< G > \mid G$ is an undirected graph with a complete subgraph of order $\frac{|V(G)|}{2}\}$ is NP-complete.
Answer:
HALF-CLIQUE is clearly in NP: a certificate is a subgraph, so a verifier will check whether the subgraph is complete and contains half the vertices of G, which is polynomial.
We then do a reduction from CLIQUE. Let (G, k) be an input for CLIQUE, and let n be the order of G. We define the following reduction:

If $k = \frac{n}{2}$: $f(G, k) = G$;

If $k < \frac{n}{2}$: $f(G, k) = H$ where H is the graph constructed by adding to G the complete graph K_{n-2k} and all the edges between it and G;

If $k > \frac{n}{2}$: $f(G, k) = H$ where H is the graph constructed by adding to G the stable graph I_{n-2k} and no edge between it and G.

It is easy to check that f is polynomial, and that there is a clique of order k in G if and only if there is a clique of order $\frac{n}{2}$ in $f(G)$. Thus, CLIQUE $\leq_p$ HALF-CLIQUE, which proves that HALF-CLIQUE is NP-complete.

□

Exercise 8 : *Two cliques*.
Prove that TWO-CLIQUES= $\{< G, k > \mid G$ is an undirected graph with two disjoint cliques of order at least $k\}$ is NP-complete.
Answer:
TWO-CLIQUES is clearly in NP: a certificate is two subsets of vertices, so a verifier will check that the two subsets are disjoint cliques, which can be done in polynomial time.
We then do a reduction from CLIQUE. Let (G, k) be an input for CLIQUE, we define the following reduction: $f(G, k) = (G + G, k)$ where $G + G$ is the graph made from two disjoint copies of G. The reduction is polynomial, and it is trivial to see that if there is a clique of size at least k in G, then there are two disjoint cliques of size at least k in $G+G$ (take the same vertices in both copies of G). Conversely, assume that there are two cliques of size at least k in $G + G$ (note that they can be in the same copy!), then each of the cliques has all its vertices in the same copy. Take the same vertices in G, and we get a clique of size at least k. Thus, the reduction is correct and CLIQUE $\leq_p$ TWO-CLIQUES. This implies that TWO-CLIQUES is NP-complete.

□

Exercise 9 : *Stable*.
Prove that STABLE= $\{< G, k > \mid G$ has an independent subgraph of order at least $k\}$ is NP-complete.
Answer:
STABLE is clearly in NP: a certificate is a subset of vertices, and a verifier will check whether it is independent and of order at least k, which is polynomial.
We then do a reduction from CLIQUE. Let (G, k) be an input for CLIQUE with n being the order of G, we define the reduction as follows: $f(G, k) = (\overline{G}, k)$ (where $\overline{G}$ is the graph on the same vertices as G and where $e \in E(\overline{G}) \Leftrightarrow e \notin E(G)$). The reduction is obviously polynomial, and it is easy to see that there is a clique of order k in G if and only if there is an independent set of order k in $\overline{G}$ (in both directions, take the same set of vertices). This implies that CLIQUE $\leq_p$ STABLE and thus that STABLE is NP-complete.

□

Exercise 10 : *Subgraphs*.
Prove that SUBGRAPH-ISOMORPHISM $= \{< G, H > \mid H$ is a subgraph of $G\}$ is NP-complete.
Answer:
SUBGRAPH-ISOMORPHISM is clearly in NP, since a certificate is a subset S of vertices of G, so a verifier will simply check if $G[S]$ is isomorphic to H, which can be done in polynomial time. Furthermore, CLIQUE $\leq_p$ SUBGRAPH-ISOMORPHISM: from an input (G, k) for CLIQUE, we can construct (G, K_k) an input for SUBGRAPH-ISOMORPHISM. The reduction is obviously polynomial and correct, and as such SUBGRAPH-ISOMORPHISM is NP-complete.

□

Exercise 11 : *0-1-matrices*.
Let M be a square matrix that has values in $\{0, 1\}$. We say that M is *simplifiable* if it is possible to transform 1's to 0's such that every row and column of M contains exactly one 1.
Prove that SIMPLIFIABLE $= \{M \mid M$ is a simplifiable matrix$\}$ is NP-complete.
Answer:
SIMPLIFIABLE is clearly in NP: a certificate is a list of the 1's that can be switched to 0's, so a verifier only needs to switch the 1's and verify that each row and each column contains exactly one 1.
We then do a reduction from UHAMPATH. Let G be an undirected graph, we let $f(G)$ be the adjacency matrix of G. It is obvious that this reduction is polynomial. Furthermore, it is easy to see that there is a hamiltonian path in G if and only if $f(G)$ is simplifiable: switch all the 1's corresponding to the edges that are not in the hamiltonian path and the matrix will have exactly one 1 in each row and column; and conversely use the edges that correspond to the 1's that are not switched and you will get a hamiltonian path.
This proves that UHAMPATH $\leq_p$ SIMPLIFIABLE, and thus that SIMPLIFIABLE is NP-complete.

□

Exercise 12 : *Domination*.
Prove that DOMINATING-SET= $\{< G, k > \mid G$ has a dominating set of size at most $k\}$ is NP-complete.
Answer:
DOMINATING-SET is clearly in NP: a certificate is a subset D of vertices, so a verifier will simply check that D is dominating and that $|D| \leq k$.
We then do a reduction from VERTEX-COVER. Let (G, k) be an input for VERTEX-COVER. We create the following graph G': $V(G') = V(G) \cup \{x_{uv} \mid uv \in E(G)\}$ and $E(G') = E(G) \cup \{ux_{uv}, vx_{uv} \mid uv \in E(G)\}$ (in other words, we replace every edge of G by a triangle). Now, we let $f(G, k) = (G', k)$. The reduction is polynomial, now let us prove its correctness.
First, note that a vertex-cover set is a dominating set, so the first side of the reduction is trivial. Conversely, assume that there is a dominating set S of size at most k in G'. Note that, if some x_{uv} is in S, then it can be replaced by either u or v and we will still have a dominating set of the same size (since the only vertices dominated by x_{uv} are itself, u and v, so they still would be dominated). Thus, we can assume that S does not contain any x_{uv}. Now, every edge is covered by S (since if an edge uv was not covered, then x_{uv} would not be dominated, a contradiction), and thus S is a vertex-cover set of G of size at most k. Thus, VERTEX-COVER $\leq_p$ DOMINATING-SET.
This implies that DOMINATING-SET is NP-complete.

□

Exercise 13 : $\neq$-assignments.
For a given boolean formula in 3-CNF, a $\neq$-assignment of its variables is an assignment where, in every clause, there are at least two literals with unequal truth values. Equivalently, a $\neq$-assignment satisfies ϕ without having all three litterals set as `true` in any clause.

1. Prove that the negation of a $\neq$-assignment is also a $\neq$-assignment.
2. Let $\neq$-3-SAT be the language of all boolean formulas in 3-CNF that have a $\neq$-assignment. Find a reduction of 3-SAT to $\neq$-3-SAT.
3. Prove that $\neq$-3-SAT is NP-complete.

Answer:

1. *In a $\neq$-assignment for ϕ, all the clauses have value* **true**, *but at least one litteral in each clause has value* **false**. *Thus, if we take the negation, there is one litteral in each clause that has value* **true**, *which implies that all clauses have value* **true**. *Furthermore, the litterals that had value* **true** *(of which there is at least one for each clause) now have value* **false**. *Thus, the negation is also a $\neq$-assignment for ϕ.*

2. *Let $\phi(x_1, \ldots, x_n)$ be a boolean formula in 3-CNF that contains ℓ clauses. We define the following reduction: for each clause, we create a variable y_i, and we have a new global variable z; each clause $C_a = (x_i \vee x_j \vee x_k)$ (w.l.o.g.) is transformed into two new clauses $(x_i \vee x_j \vee y_a) \wedge (\overline{y_a} \vee x_k \vee z)$. Let us call $f(\phi(x_1, \ldots, x_n)) = \phi'(x_1, \ldots, x_n, y_1, \ldots, y_\ell, z)$ the 3-CNF constructed this way. This reduction is polynomial, now let us prove that it is correct.*

   First, assume that there is an assignment satisfying ϕ, then we set the values of x_i's in ϕ' as the same than in ϕ, and we set $z =$ **false**. *Now, we must make sure that at least one litteral has value* **false** *in each clause. Let $C_a = (x_i \vee x_j \vee x_k)$ (w.l.o.g.), then we have the two clauses $(x_i \vee x_j \vee y_a) \wedge (\overline{y_a} \vee x_k \vee z)$. If both x_i and x_j have value* **false**, *then we set y_a as* **true**; *otherwise we set it as* **false**: *the first clause has value* **true** *and at least one of its litterals has value* **false**. *Furthermore, since $z =$* **false**, *the second clause has at least one litteral with value* **false**, *and it will also be verified. This is depicted in the following table:*

Value of (x_i, x_j)	*Value of y_a*	*Value of x_k*
true *and* **true**	**false**	*no impact since $\overline{y_a} =$* **true** *and $z =$* **false**
true *and* **false**	**false**	*no impact since $\overline{y_a} =$* **true** *and $z =$* **false**
false *and* **false**	**true**	**true** *since C_a has value* **true**

   Thus, we have a $\neq$-assignment for ϕ'.

   Now, assume that there is a $\neq$-assignment for ϕ'. We can assume that $z =$ **false**, *since if it is not the case then we can take the opposite assignement, which is also a $\neq$-assignment by the first question. For each clause C_a, if $y_a =$* **true** *then we necessarily have $x_k =$* **true**; *and otherwise we necessarily have at least one of x_i or x_j with value* **true** *(since otherwise, one of the two clauses created from C_a would have value* **false**). *Thus, in this assignment, for every clause, at least one litteral has value* **true**, *and thus the same assignment for the x_i's is a satisfying assignment for ϕ.*

   The reduction being correct, we have 3-SAT $\leq_p$ $\neq$-3-SAT.

3. *$\neq$-3-SAT is clearly in NP: a certificate is an assignment, so a verifier will have to check whether it is a $\neq$-assignment (checking that every clause has value* **true** *and contains at least one litteral with value* **false**), *which is polynomial. Furthermore, we know by the previous discussion that 3-SAT $\leq_p$ $\neq$-3-SAT. Since 3-SAT is NP-complete, this implies that $\neq$-3-SAT is NP-complete.*

□

Exercise 14 : *Cutting as much as we can.*
Prove that MAX-CUT= $\{< G, k > \mid G$ has a cut of size at least $k\}$ is NP-complete (hint: reduce from $\neq$-3-SAT, G may contain multiedges).
Answer:

MAX-CUT is clearly in NP: a certificate is a subset S of vertices, so a verifier only has to check that there are at least k edges between vertices in S and vertices in $V(G) \setminus S$, which can be done in polynomial time.

We then do a reduction from $\neq$-3-SAT. Let $\phi(x_1, \ldots, x_n)$ be a boolean formula in 3-CNF, and let ℓ be the number of clauses in ϕ. We construct the graph G the following way: we create two vertices v_i and $\overline{v_i}$ for every variable x_i; then for each clause we link the three vertices associated with its litterals as a triangle (so if, for example, $C_a = (x_i \vee x_j \vee x_k)$, we have the edges v_iv_j, v_iv_k and v_jv_k); finally, for each variable x_i, let m_i be the number of times it appears (either as x_i or as $\overline{x_i}$) in a clause, we add m_i multiedges between v_i and $\overline{v_i}$. We let $f(\phi) = (G, 5\ell)$. This reduction is polynomial, let us prove its correctness.

First, assume that there is a $\neq$-assignment for ϕ. We set $S = \{v_i \mid x_i = \texttt{true}\} \cup \{\overline{v_i} \mid x_i = \texttt{false}\}$ (that is, the vertices corresponding to the litterals with value `true`). Now, let us count the edges crossing between S and $V(G) \setminus S$. On the first hand, all the multiedges between v_i's and $\overline{v_i}$'s will be counted, and there are 3ℓ such multiedges (since there are ℓ clauses of 3 litterals). On the other hand, every clause contains at least one litteral with value `true` and one litteral with value `false`, so for every clause there are exactly two edges from the triangle that cross from S to $V(G) \setminus S$ (since one (resp. two) litterals are in S and two (resp. one) are not in S), so there are 2ℓ such edges between S and $V(G) \setminus S$. In total, there are 5ℓ edges between S and $V(G) \setminus S$, which is what we wanted.

Conversely, assume that there is a cut with at least 5ℓ edges crossing it. Note that, if both v_i and $\overline{v_i}$ are on the same side of the cut, then we can put them on different sides without decreasing the number of edges crossing the cut. Indeed, assume that there are two vertices v_i and $\overline{v_i}$ that are on the same side of the cut, assume that x_i is in a clauses and that $\overline{x_i}$ is in b clauses, so there are m_i multiedges between v_i and $\overline{v_i}$, and $a+b$ edges incident with v_i and $\overline{v_i}$ that cross the cut; furthermore we have $a+b = 2m_i$. Putting v_i on the other side of the cut would let us have $b+m_i$ edges crossing it; and putting $\overline{v_i}$ would let us have $a+m_i$ edges crossing it. If doing both did decrease the number of edges crossing the cut, then we would have both $a + m_i < a + b$ and $b + m_i < a + b$, thus we would have $a + b + 2m_i < 2(a + b)$, that is, $2m_i < a + b$, a contradiction (since $a + b = 2m_i$). Thus, we can assume that v_i and $\overline{v_i}$ are on opposite sides of the cut. This implies that the 3ℓ multiedges contribute to the cut, and thus that the clause gadgets contribute to at least 2ℓ to the cut. This implies that, for every triangle (clause gadget), two edges contribute (since it is impossible to have all three edges from a triangle crossing a cut). By setting the vertices on one side of the cut as `true` and the others as `false`, we have a $\neq$-assignment for ϕ.

Thus, $\neq$-3-SAT $\leq_p$ MAX-CUT, which implies that MAX-CUT is NP-complete.

□

Exercise 15 : *Cutting exactly as much as we need.*

Prove that EXACT-CUT= $\{< G, k > \mid G$ has a cut of size exactly $k\}$ is NP-complete.

Answer:

EXACT-CUT is obviously in NP: the argument is the same than for MAX-CUT. Now, we do a reduction from MAX-CUT.

Let (G, k) be an input for MAX-CUT and let $e = |E(G)|$. We create the graph G' as the disjoint union of G and a star $K_{1,e}$, and let $f(G, k) = (G', k + e)$. This reduction is clearly polynomial. Furthermore, if there is a cut of size $\ell \geq k$ in G, then we can find a cut of size $k + e$ in G': take the same cut in the copy of G, and take $k + e - \ell$ edges in the star by putting the center and $\ell - k$ leaves on one side of the cut and $k + e - \ell$ leaves on the other side (this is possible since $\ell \geq k$). Conversely, if there is a cut of size exactly $k + e$ in G', then there are at most e edges crossing the cut that are in the star, and as such there are at least k edges crossing the cut in G, and so we have a cut of size at least k in G. This proves that MAX-CUT $\leq_p$ EXACT-CUT.

Altogether, this proves that EXACT-CUT is NP-complete.

□

Exercise 16 : *3-color.*

Prove that 3-COLOR= $\{< G > \mid G$ is an undirected graph that has a 3-colouring$\}$ is NP-complete.

Answer:

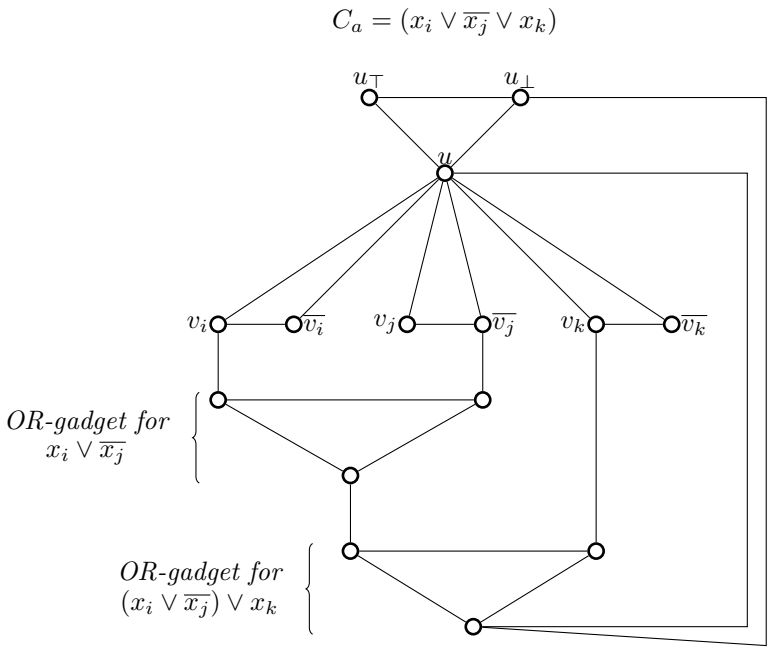
3-COLOR is clearly in NP: a certificate is an assignment of colours 1, 2 or 3 to the vertices, so a verifier will check that every vertex is coloured and that no two adjacent vertices have the same colour, which is polynomial.

We then do a reduction from 3-SAT, so we will construct a graph from a boolean formula. The idea is to

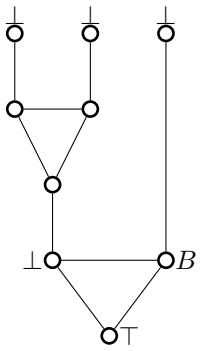
have a "palette" gadget that is a triangle. One vertex, called $u_\top$, will be used to find which variables will be assigned the value **`true`**; *another vertex, called $u_\bot$, will be used to find which variables will be assigned the value* **`false`**.
Let $\phi(x_1, \ldots, x_n)$ be a boolean formula in CNF with clauses of size 3. We create the graph G as follows:

1. *Create a triangle with vertices $u_\top$, $u_\bot$ and u;*
2. *For every variable x_i, create two vertices v_i and $\overline{v_i}$, and link both these vertices to u;*
3. *For every clause C_a, use OR-gadgets to link its three variables, and add an edge between the exit of the gadgets and $u_\bot$ and u.*

This is depicted in the following picture:



It is easy to see that this reduction is polynomial, since we create $2n + 6m + 3$ vertices (where n is the number of variables and m the number of clauses in ϕ), and thus a polynomial number of edges. Now, we need to prove that ϕ is satisfiable if and only if G is 3-colourable. We will call the three colours $\top$, $\bot$ and B.
First, assume that ϕ is satisfiable. We colour the graph the following way: if x_i has the value **`true`**, *then colour v_i with $\top$ and $\overline{v_i}$ with $\bot$; also colour $u_\top$ with $\top$, $u_\bot$ with $\bot$ and u with B. Note that, since at least one of the three variables of a clause is* **`true`**, *we can colour the end vertex of the OR-gadgets with $\top$. The other vertices within the gadget can then be coloured without creating any conflict.*
Conversely, assume that there is a 3-colouring of G. Without loss of generality, assume that $u_\top$ has colour $\top$, $u_\bot$ has colour $\bot$, and u has colour B (if this is not the case, we rename the colours). Note that all vertices v_i and $\overline{v_i}$ have colours in $\{\top, \bot\}$ and that v_i and $\overline{v_i}$ cannot have the same colour. Thus, we assign the value **`true`** *to x_i if and only if v_i is coloured with $\top$. Now, we prove that this assignment satisfies ϕ. Assume by contradiction that there is a clause $C_a = (x_i \vee x_j \vee x_k)$ (without loss of generality) that is not satisfied. This means that v_i, v_j and v_k have colour $\bot$. However, the end vertex of the OR-gadget for C_a has colour $\top$ (since it is linked with $u_\bot$ and u), which implies that we have the following colour assignment:*



As we can see, the intermediary OR-gadget for C_a is a triangle, but all its vertices are neighbours of a vertex coloured with $\perp$, a contradiction. Thus, every clause is satisfied, and thus ϕ is satisfied.
Altogether, this proves that 3-SAT $\leq_p$ 3-COLOR, and thus 3-COLOR is NP-complete.

□

Exercise 17 : *Clique in a restricted family.*
Prove that REG-CLIQUE $= \{< G, k > \mid G$ is a regular undirected graph with a clique of order at least $k\}$ is NP-complete.
Answer:
Since CLIQUE is in NP, REG-CLIQUE is also in NP. We then do a reduction from CLIQUE. First, note that if $k \in \{1, 2\}$ then the language is trivial (if $k = 1$ then we just need to check that there is a vertex, and if $k = 2$ then we just need to check that there is an edge), so if $k \in \{1, 2\}$ then the reduction consists in determining if G contains either a vertex or an edge, and then creating a regular graph such that the answer is the same (for $k = 1$ either a graph with no vertex or with a single vertex, for $k = 2$ either a graph with no vertex or K_2). In the following, we will assume that $k \geq 3$.
Let (G, k) be an input for clique, and let Δ be the maximum degree of G. We construct the following graph G': first, we create Δ disjoint copies of G, noted $G_1, G_2, \ldots, G_\Delta$; then, for every vertex v such that $d(v) < \Delta$, we create $\Delta - d(v)$ supplementary vertices $a_v^1, \ldots, a_v^{\Delta - d(v)}$ that are linked to every copy of v in $G_1, \ldots, G_\Delta$. It is easy to see that G' is Δ-regular: every vertex v in a copy of G has $d(v)$ neighbours in its copy, and $\Delta - d(v)$ neighbours (the a_v^i); and $d(a_v^i) = \Delta$ since it has one neighbour in each of the Δ copies of G. Thus, we have the following reduction: $f(G, k) = (G', k)$. It is easy to see that the reduction is polynomial, since $\Delta = O(n)$ and as such G' contains $O(n^2)$ vertices (and thus a polynomial number of edges).
We now prove that the reduction is correct. First, assume that G has a clique of order at least k. Then, by taking the same vertices in any G_i, it is easy to see that G' has a clique of order at least k. Conversely, assume that G' has a clique of order at least k. Since all the G_i's are disjoint, the clique cannot contains vertices from different G_i's. Furthermore, assume that there is a vertex a_v^i that belongs to this clique; then since $k \geq 3$ there are at least two other vertices in the clique, however all neighbours of a_v^i are disjoint copies of v, a contradiction. Thus, the clique is contained in a single G_i. By taking the same vertices, there is a clique of order at least k in G. Thus, our reduction is correct and CLIQUE $\leq_p$ REG-CLIQUE. This implies that REG-CLIQUE is NP-complete.

□

Exercise 18 : *A small break.*
This is not an exercise. Give a read to the paper *Minesweeper is NP-complete*, available here:

`http://www.minesweeper.info/articles/MinesweeperIsNPComplete.pdf`

□