TD6 – Réductions, NP-difficulté, NP-complétude

Rappel 1 : Une reduction many-one polynomiale de L_1 à L_2, est donnée par un algo f en temps poly qui transforme chaque mot sur L_1 en un mot sur L_2, et préserve l'acceptation :

$$x \in L_1 \iff f(x) \in L_2.$$

On note alors $L_1 \leq_m^p L_2$, car le problème L_2 est *au moins aussi difficile* que le problème L_1.

Rappel 2 : L_2 est NP-difficile si et seulement si pour tout $L_1 \in \mathsf{NP}$ on a $L_1 \leq_m^p L_2$.
Rappel 3 : L_2 est NP-complet si et seulement si $L_2 \in \mathsf{NP}$ et L_2 est NP-difficile.
Rappel 4 : Si L_1 est NP-difficile et $L_1 \leq_m^p L_2$ alors L_2 est NP-difficile.

Donc, pour répondre à un exercice de la forme

« montrer que le problème **Toto** est NP-complet »,

on pourra remplir le texte à trou suivant :

(a) **Toto** $\in \mathsf{NP}$, car ....
....
⟨*ici soit on donne un algo dans* NP *pour décider* **Toto***, soit on utilise la char. exist. de* NP⟩ .....
....

(b) Pour le problème **Tata** que l'on sait déjà être NP-difficile, on a **Tata** $\leq_m^p$ **Toto**, car il existe la transformation $f : \Sigma^*_{\textbf{Tata}} \to \Sigma^*_{\textbf{Toto}}$ définie par ....
....
⟨*ici on explique comment transformer les instances de* **Tata** *en des instances de* **Toto**⟩ ....
....
...., qui est :

- **i.** calculable en temps polynomial, car ....
  ....
  ⟨*ici on peut en général argumenter simplement : objets de taille poly faciles à générer…*⟩ ....
  ....
- **ii.** et telle que, pour tout $x \in \Sigma^*_{\textbf{Tata}}$ on a $x \in \textbf{Tata} \iff f(x) \in \textbf{Toto}$. En effet :
  - i. pour tout $x \in \Sigma^*_{\textbf{Tata}}$ on a $x \in \textbf{Tata} \implies f(x) \in \textbf{Toto}$, car ....
    ....
    ....
    ⟨*ici se trouve le cœur de la démonstration – ventricule gauche*⟩ ....
    ....
    ....
    ....
  - ii. pour tout $x \in \Sigma^*_{\textbf{Tata}}$ on a $f(x) \in \textbf{Toto} \implies x \in \textbf{Tata}$, car ....
    ....
    ....
    ⟨*ici se trouve le cœur de la démonstration – ventricule droit*⟩ ....
    ....
    ....
    ....

Donc le problème **Toto** est NP-complet.

Exercice 1. *Problèmes* NP*-complets*

Les définitions des problèmes sont données ci-après. On supposera acquis que

3-SAT, **Clique** et **Couverture par Sommets** sont NP-complets.

Pour répondre à une question on pourra supposer que l'on a déjà répondu aux précédentes.

1. Montrer que **Isomorphisme de Sous-Graphes** est NP-complet. *Indice : réduire depuis* **Clique**.
2. Montrer que **Ensemble Dominant** est NP-complet. *Indice : réduire depuis* **Couverture par Sommets (Node Cover)**.
3. Montrer que **3-Colorabilité** est NP-complet. *Indice : réduire depuis* **3-SAT**.
4. Montrer que **Cycle Hamiltonien** est NP-complet. *Indice : réduire depuis* **3-SAT**.

3-SAT
entrée : une formule propositionnelle ϕ en forme normale conjonctive, dont toutes les clauses sont de taille exactement trois.
question : y a-t-il une affectation qui satisfait ϕ ?

Clique
entrée : un graphe non-orienté $G = (V, E)$ et un entier $k \in \mathbb{N}$.
question : G contient-il une clique de taille k ?

Couverture par Sommets (Node Cover)
entrée : un graphe non orienté $G = (S, A)$ et un entier naturel k.
question : existe-t-il un sous-ensemble de sommets $C \subseteq S$ de taille au plus k tel que chaque arête de G a au moins une extrémité dans C ?

Isomorphisme de Sous-Graphes
entrée : deux graphes orientés $G = (S, A)$ et $H = (S', A')$.
question : G possède-t-il un sous-graphe isomorphe à H ?

Ensemble Dominant
entrée : un graphe non orienté $G = (S, A)$ et un entier naturel k.
question : existe-t-il un sous-ensemble de sommets $D \subseteq S$ de taille au plus k tel que chaque sommet de G est soit lui-même dans D, soit il est adjacent à un sommet dans D ?

3-Colorabilité
entrée : un graphe non orienté $G = (S, A)$.
question : existe-t-il une coloration des sommets de G avec au plus 3 couleurs telle que deux sommets adjacent soient toujours de deux couleurs différentes ?

Cycle Hamiltonien
entrée : un graphe non orienté $G = (S, A)$.
question : existe-t-il un cycle dans G qui qui passe par tous les sommets une fois et une seule ?

Solution 1.1 Voici un algorithme non déterministe pour vérifier si $G = (S, A)$ admet un sous-graphe isomorphe à $H = (S', A')$, c'est-à-dire, s'il existe une fonction injective $\varphi\colon S' \to S$ tel que, pour chaque arête $\{i, j\} \in A'$, on a une arête correspondante $\{\varphi(i), \varphi(j)\} \in A$. Cela correspond à dire que, en éliminant une partie des sommets et des arêtes de G, on peut obtenir un graphe ayant la même forme que H. On assume une représentation par matrices d'adjacence des deux graphes.

```
algorithme iso-sous-graphes(G, H):
    phi := tableau d'entiers de longueur H.n
    pour i := 1 à H.n faire
        phi[i] := deviner(1, ..., G.n)
    pour i := 1 à H.n faire
        pour j := i+1 à H.n faire
            si phi[i] = phi[j] alors
                rejeter
    pour i := 1 à H.n faire
        pour j := 1 à H.n faire
            si H.A[i][j] et non G.A[phi[i]][phi[j]] alors
                rejeter
    accepter
```

Cet algorithme représente la fonction injective φ par un tableau `phi` tel que $\varphi(i) =$ `phi[i]`. D'abord on devine l'image $\varphi(i)$ de chaque sommet i de H, c'est-à-dire, un sommet de G. Cela prend du temps $\Theta(n)$, ou n est le nombre de sommets de H. Puis on vérifie qu'il s'agit bien d'une fonction injective, c'est-à-dire, que chaque case de `phi` contient une valeur différente ; cela prend du temps $\Theta(n^2)$. Enfin, on vérifie que chaque arête $\{i, j\}$ de H a une arête correspondante $\{\varphi(i), \varphi(j)\}$ de G ; si ce n'est pas le cas, on rejette. Cela prend également du temps $\Theta(n^2)$. Si toutes les arêtes nécessaires existent, on a trouvé un sous-graphe de G isomorphe à H et on accepte. En total, cet algorithme prends du temps $\Theta(n^2)$. Donc le problème appartient à NP.

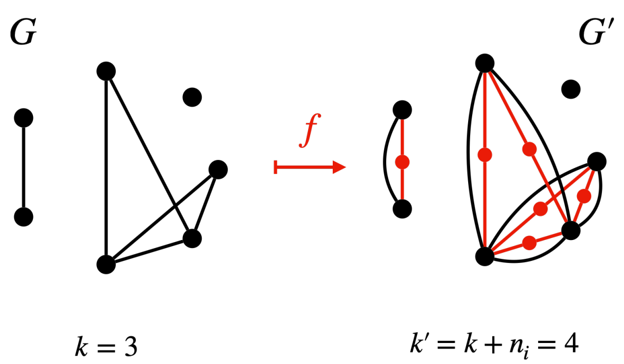
Pour montrer sa NP-completude, on décrit une réduction depuis **Clique**. Un graphe non-orienté $G = (S, A)$ a une clique de taille k si et seulement si il contient un sous-graphe isomorphe au graphe complet (ayant toutes les arêtes) de k sommets. Donc une réduction appropriée est $f(G, k) = (G, H)$, où H est le graphe complet de k sommets. Cette réduction peut-être calculé en temps polynomial, puisque il s'agit de recopier en sortie le graphe G et d'y ajouter le graphe H, dont on peut construire en temps $O(k^2)$ la matrice d'adjacence.

Solution 1.2 Voici un algorithme non-déterministe pour le problème :

```
1   algorithme ensemble-dominant(G, k):
2       D := tableau de longueur G.n
3       pour i := 1 à G.n faire
4           D[i] := deviner(vrai, faux)
5       c := 0
6       pour i := 1 à G.n faire
7           si D[i] alors
8               c := c + 1
9       si c > k alors
10          rejeter
11      pour i := 1 à G.n faire
12          si D[i] = faux alors
13              trouvé := faux
14              pour j := 1 à G.n faire
15                  si G.A[i][j] et D[j] alors
16                      trouvé := vrai
17              si non trouvé alors
18                  rejeter
19      accepter
```

On devine un tableau `D` de booléens qui correspond au sous-ensemble $D \subseteq S$ (lignes 2–4) et on vérifie qu'il contient au plus k sommets (lignes 5–9), en rejetant si ce n'est pas le cas. Cela prend du temps $\Theta(n)$. Ensuite, on vérifie si chaque sommet i du graphe est soit lui-même dans D (lignes 11-12), soit adjacent à un sommet dans D (lignes 13–18), en rejetant si ce n'est pas le cas. Cela prend tu temps $\Theta(n^2)$ dans le pire des cas. Si chaque sommet de G est « dominé » par D, alors on accepte (ligne 19). Le temps de calcul total est de $\Theta(n^2)$, ce qui prouve que le problème appartient à NP.

On montre que le problème est NP-complet par réduction depuis **Couverture par Sommets**. Étant donné un graphe non orienté $G = (S, A)$ et un entier k, on construit un deuxième graphe $G' = (S', A')$ et un entier k' en ajoutant un sommet intermédiaire pour chaque arête de G, de la forme suivante :



Donc on a $S' = S \cup \{ij : \{i,j\} \in A\}$, où les ij sont les sommets intermédiaires ajoutés et $A' = A \cup \big\{\{i, ij\}, \{ij, j\} : \{i,j\} \in A\big\}$. La valeur de k' est donné par k plus le nombre n_i de sommets isolés (c'est-à-dire, qui ne sont pas reliés à d'autres sommets) dans G. La transformation peut-être calculée en temps polynomial, vu que le nombre de sommet et d'arêtes ajoutés sont $|A|$

et $2 \times |A|$ respectivement, et le calcul de k' ne comporte que la recherche des sommets isolés et une addition.

Si G admet une couverture par sommets $C \subseteq S$ de taille au plus k, soit $D = S \cup \{i : i \text{ est isolé dans } G\}$. Alors D a taille au plus k' (certains sommets isolés pourraient déjà appartenir à C). Montrons que D est un ensemble dominant :

— tous les sommets isolés appartiennent à D ;
— pour chaque arête $\{i, j\} \in A$, on a soit $i \in C$, soit $j \in C$ (soit les deux) ; donc l'un des sommets appartient à D aussi et l'autre, même s'il n'appartient pas à D lui-même, est relié à l'autre ;
— chaque nouveau sommet ij est relié à i et j qui sont reliés entre eux ; donc soit i, soit j appartient à C et donc à D, et par conséquent ij est relié à un sommet dans D.

Donc, chaque sommet de G' est soit dans D, soit relié à un sommet dans D.

Vice-versa, supposons que G' possède un ensemble dominant D de taille $k' = k + n_i$. Alors D contient tous les sommets isolés de G' (qui sont les mêmes qu'en G). Considérons l'ensemble $C' = D - \{i : i \text{ est isolé}\}$, donc $|C'| = k' - n_i = k$. Or, C' pourrait contenir des nouveaux sommets de la forme ij. Un sommet de ce type se domine lui-même, et en plus il domine i et j ; donc on peut le remplacer dans C' par i et les trois sommets restent dominés. Soit donc C l'ensemble obtenu à partir de C' en remplaçant chaque sommet du type ij par i. Alors C ne contient que des sommets de G et, en plus, on a $|C| \leq k$ aussi (on peut réduire la taille par rapport à C' si, en remplaçant ij par i, on avait déjà $i \in C$). Montrons que C est une couverture par sommets de G : si, par l'absurde, il existait une arête $\{i, j\} \in A$ telle que ni $i \in C$, ni $j \in C$, alors le sommet ij de G' ne serait pas adjacent à un sommet de C', ce qui est une contradiction puisque C' était un ensemble dominant.

Cela montre que G a une couverture par sommets de taille k si et seulement si G' a un ensemble dominant de taille k'. Donc la transformation de (G, k) en (G', k') est bien une réduction many-one polynomiale, et le problème **Ensemble Dominant** est NP-complet.

Solution 1.3 L'algorithme non-déterministe suivant montre que **3-Colorabilité** est NP-complet. On assume une représentation en matrice d'adjacence pour le graphe G.

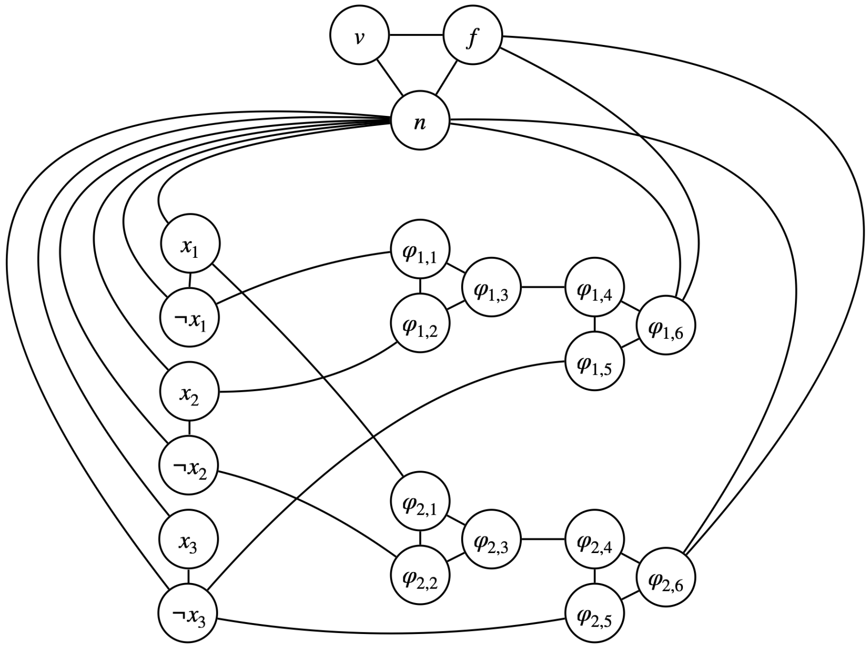
```
algorithme 3-colorabilité(G):
    couleur := tableau de longueur G.n
    pour i := 1 à G.n faire
        couleur[i] := deviner(rouge, bleu, vert)
    pour i := 1 à G.n faire
        pour i := 1 à G.n faire
            si G.A[i][j] et couleur[i] = couleur[j] alors
                rejeter
    accepter
```

L'algorithme devine une couleur parmi trois pour chaque sommet, ce qui prend du temps $\Theta(n)$, n étant le nombre de sommets. Ensuite, on traverse la matrice d'adjacence en temps $\Theta(n^2)$ en vérifiant s'il existe une arête ayant les extrémités de la même couleur ; si c'est le cas, la coloration deviné n'est pas correcte. Si, en revanche, chaque arête a les extrémités de couleurs différentes, on peut conclure que le graphe admets une 3-coloration et accepter. Le temps de calcul total est $\Theta(n^2)$.

Pour montrer que le problème est NP-complet, on y réduit le problème **3-SAT**. Soit φ une formule booléenne en forme normale conjonctive avec exactement 3 littéraux par clause, par exemple la formule suivante définie sur $n = 3$ variables (x_1, x_2, x_3) et $m = 2$ clauses (φ_1, φ_2) :

$$\varphi = \underbrace{(\neg x_1 \vee x_2 \vee \neg x_3)}_{\varphi_1} \wedge \underbrace{(x_1 \vee \neg x_2 \vee \neg x_3)}_{\varphi_2}$$

On construit un graphe orienté correspondent $G = (S, A)$ de la forme suivante :



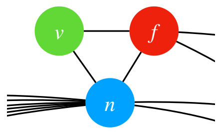
Les sommets du graphe comprennent tous les littéraux de la forme x_i et $\neg x_i$ pour chaque variable x_i de la formule (y compris les littéraux qui n'apparaissent pas dans la formule, par exemple x_3 pour φ. On a aussi six sommets $\varphi_{j,1}, \ldots, \varphi_{j,6}$ pour chaque clause φ_j de la formule. Enfin, on a trois sommets v, f, n. Le nombre de sommets du graphe est donc $2n + 6m + 3$, où n est le nombre de variables et m le nombre de clauses de φ, ce qui est polynomial (plus précisément, linéaire) par rapport à la taille de la formule.

Les arêtes sont définie de la forme suivante : les sommets v, f, n sont relié en triangle. Pour chaque variable x_i, les sommets $n, x_i, \neg x_i$ sont également reliés en triangle. Pour chaque clause φ_j, les sommets $\varphi_{j,1}, \varphi_{j,2}, \varphi_{j,3}$ et les sommets $\varphi_{j,4}, \varphi_{j,5}, \varphi_{j,6}$ sont reliés en triangles, et les sommets $\varphi_{j,3}$ et $\varphi_{j,4}$ sont également reliés. D'autres triangles sont formés par les sommets $\varphi_{j,6}, f, n$. Enfin, pour chaque clause φ_j, les trois noeuds correspondants aux littéraux de la clause φ_j sont reliés, respectivement, aux sommets $\varphi_{j,1}$, $\varphi_{j,2}$ et $\varphi_{j,5}$.

Comme le nombre de sommets du graphe G est polynomial, et que la structure du graphe est régulière (il s'agit de construire une série de triangles et d'ajouter les arcs manquants), il est possible de le construire à partir de φ en temps polynomial.

Il faut maintenant montrer que le graphe G admets une 3-coloration si et seulement si la formule φ est satisfaisable.

Observons d'abord que les trois sommets v, f, n sont relié en triangle et ils doivent donc avoir trois couleurs différentes. Supposons, sans perte de généralité, qu'on colorie v en vert, f en rouge et n en bleu :

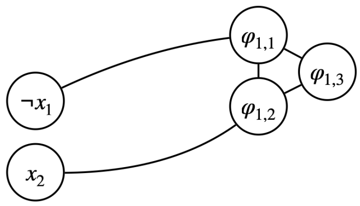


On interprète comme « vrai » la couleur de v (vert) et comme « faux » la couleur de f (rouge). (On peut évidemment choisir d'autres colorations pour ces trois sommets, en changeant de façon correspondante le raisonnement qui suit.)

Pour chaque variable x_i, les sommets x_i, $\neg x_i$ et n forment un triangle, ce qui force x_i à être vert et $\neg x_i$ à être rouge, ou bien vice-versa. Cela correspond à choisir une valeur de vérité pour chaque variable de la formule.

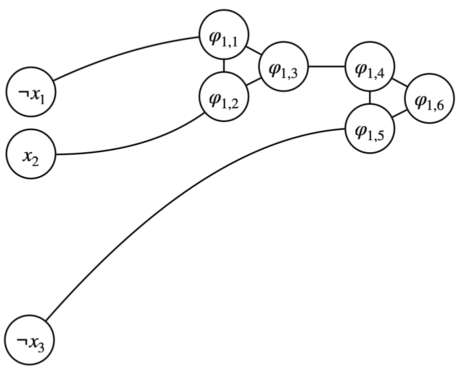
Pour chaque clause φ_j, les sommets $\varphi_{j,6}$, f et n forment un triangle, ce qui force la couleur vert pour le sommet $\varphi_{j,6}$.

Considérons maintenant le sous-graphe suivant dans le graphe correspondant à la formule φ d'exemple :



Il est possible de colorier $\varphi_{1,3}$ en vert seulement si $\varphi_{1,1}$ est bleu et $\varphi_{1,2}$ est rouge, ou vice-versa. En gardant à l'esprit le fait que $\neg x_1$ et x_2 (les premiers deux littéraux de la clause φ_1) sont nécessairement de couleur rouge ou vert, cela force l'un des deux (celui qui est adjacent au sommet $\varphi_{1,1}$ ou $\varphi_{1,2}$ qui est colorié en rouge) à être vert. Donc, $\varphi_{1,3}$ ne peut être vert que si au moins un parmi $\neg x_1$ et x_2 est vert, ou les deux le sont à la fois. Donc, le « gadget » formé par les sommets $\varphi_{1,1}$, $\varphi_{1,2}$ et $\varphi_{1,3}$ peut être interprété comme porte logique $\vee$ qui calcule la disjonction des valeurs $\neg x_1$ et x_2.

Considérons maintenant aussi le troisième littéral $\neg x_3$ de la clause φ_1, c'est-à-dire le sous-graphe suivant :



Comme discuté précédemment, le sommet $\varphi_{1,6}$ est nécessairement vert. Les sommets $\varphi_{1,4}, \varphi_{1,5}$ et $\varphi_{1,6}$ se comportent comme $\varphi_{1,1}, \varphi_{1,2}$ et $\varphi_{1,3}$, c'est-à-dire comme une porte logique $\vee$. Donc au moins un parmi $\varphi_{1,3}$ et $\neg x_3$ doit être vert et donc, globalement, au moins parmi $\neg x_1, x_2$ et $\neg x_3$ doit être vert pour pourvoir colorier $\varphi_{1,6}$ en vert. Le gadget formé par les sommets $\varphi_{1,1}, \ldots, \varphi_{1,6}$ évalue donc la clause φ_j de la formule en fonction de l'affectation des variables donné par la coloration des sommets $\neg x_1, x_2, \neg x_3$.

Le même raisonnement s'applique à tous les autres gadget formés par les sommets du type $\varphi_{j,1}, \ldots, \varphi_{j,6}$, qui correspondent aux clauses φ_j.

Si on considère donc une affectation des variables qui satisfait la formule, on peut trouver une 3-coloration correspondante dans le graphe, en choisissant la couleur vert pour les variables vraies, et la couleur rouge pour les fausses. Vice-versa, étant donnée une 3-coloration du graphe, en interprétant les sommets x_i de la même couleur du sommet v comme variables vraies et ceux de la même couleur que f comme variables fausses, on obtient une affectation qui satisfait la formule. Donc, la formule est satisfaisable si et seulement si le graphe correspondent admet une 3-coloration.

La transformation de la formule en graphe peut-être effectué en temps polynomiale ; il s'agit donc d'une réduction many-one polynomiale de **3-SAT** à **3-Coloration**. Étant **3-SAT** NP-complet, **3-Coloration** est donc NP-difficile ; comme en plus il appartient à NP, on peut conclure que **3-Coloration** est également NP-complet.

Solution 1.4 Voici un algorithme non-déterministe pour le problème :

```
1    algorithme cycle-hamiltonien(G):
2        perm := tableau d'entiers de longueur G.n
3        pour i := 1 à G.n faire
4            perm[i] := deviner(1, ..., G.n)
5        pour k := 1 à G.n faire
6            trouvé := faux
7            pour i := 1 à G.n faire
8                si perm[i] = k alors
9                    trouvé := vrai
10           si non trouvé alors
11               rejeter
12       pour i := 1 à G.n - 1 faire
13           si non G.A[perm[i]][perm[i+1]] alors
14               rejeter
15       si non G.A[perm[n]][perm[1]] alors
16           rejeter
17       accepter
```

Cet algorithme d'abord devine une possible permutation des sommets en remplissant un tableau `perm` de longueur n, le nombre de sommets (lignes 2–4). Cela prend du temps $\Theta(n)$.

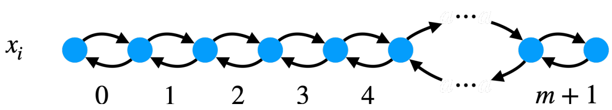
Ensuite, on vérifie qu'il s'agit bien d'une permutation, c'est-à-dire si tout les sommets k de 1 à n y apparaissent (lignes 5–10), en rejetant si ce n'est pas le cas (remarquez que, étant le tableau `perm` de longueur n, le fait que chaque entier de l'intervalle $[1, n]$ y apparaisse implique qu'il apparaît exactement une fois). Cela prend du temps $\Theta(n^2)$.

Ensuite, on vérifie (lignes 12–16) s'il est possible de parcourir les sommets dans l'ordre donné par `perm`, c'est-à-dire, si $\{$`perm[i]`, `perm[i+1]`$\}$ est bien une arête de G pour tout $i = 1, \ldots, n-1$, et si $\{$`perm[n]`, `perm[1]`$\}$ existe également (il s'agit de la dernière arête, qui permet de revenir au point de départ). Si l'une des arêtes est manquant, alors on rejette. Cela prend du temps $\Theta(n)$.

Si on a deviné une permutation telle que toutes les arêtes nécessaires apparaissent dans G, alors on accepte, puisque il s'agit d'un cycle hamiltonien (ligne 17). Le temps de calcul total est $\Theta(n^2)$, ce qui montre que le problème appartient à NP.

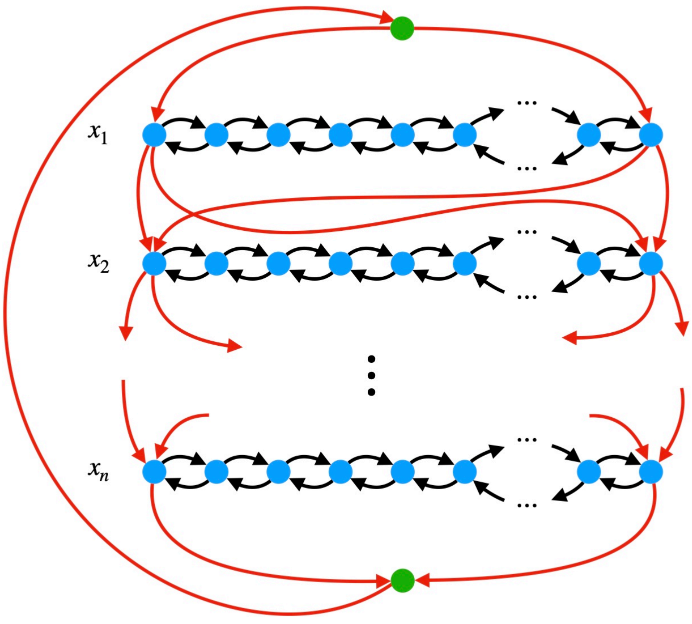
Pour montrer que le problème est NP-complète, on effectue une réduction à partir de **3-SAT**. Étant donné une formule $\varphi = \varphi_1 \wedge \varphi_2 \wedge \cdots \varphi_m$ en forme normale conjonctive avec 3 littéraux par clause, n variables et m clauses, on construit un graphe G correspondant de la façon suivante.

Pour chaque variable x_i de la formule on a un gadget de la forme :

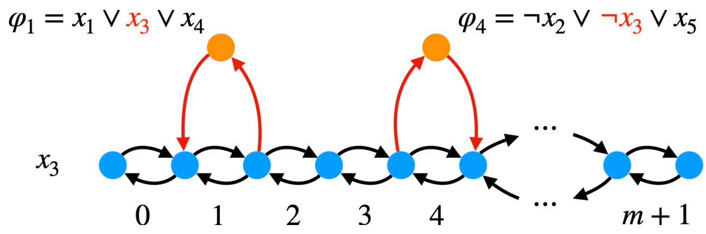


Intuitivement, parcourir ce gadget de gauche à droite correspond à choisir la valeur faux pour x_i, alors que le parcourir de droite à gauche correspond à choisir $x_i =$ vrai.

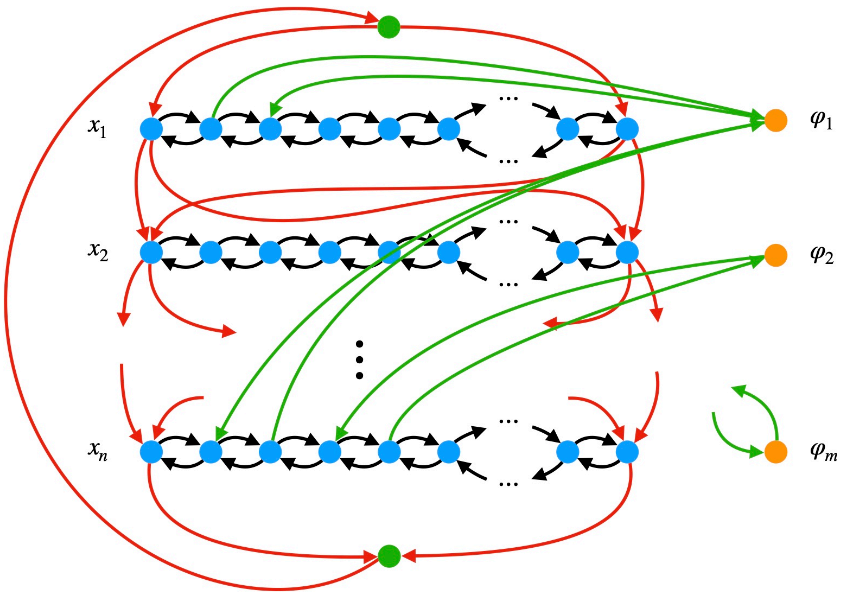
Les gadgets pour chaque variable $x_1, \ldots, x_n$ sont reliés de la forme suivante, qui permet de passer au gadget de la variable suivante dans l'une des deux directions, ainsi que de renfermer le cycle :



On ajoute également un sommet pour chaque clause φ_j et on le relié au gadget de la variable x_i (en position correspondante à la numérotation de la clause) si cette variable apparaît dans la formule. La direction des arcs est de droite à gauche si la variable apparaît de forme positive dans la clause, et de gauche à droite si elle apparaît de forme négative. Cela permet de traverser le sommet correspondant à la clause si la direction avec laquelle on parcourt le gadget de la variable correspond à une affectation qui satisfait la clause :



Globalement, chaque sommet associé à une clause est relié à trois gadget correspondants aux variables qu'y apparaissent. Cela donne un graphe final ayant une forme du type :



La taille de ce graphe est polynomiale (on a $\Theta(mn)$ sommets) et, comme les arête correspondent à la structure de la formule φ, on peut le construire en temps polynomial.

Une affectation des variables de la formule correspond à une direction de parcours pour chaque gadget correspondant à une variable. Si une telle affectation satisfait la formule, alors il sera possible de visiter chaque sommet φ_j à partir du gadget d'une variable (si plusieurs littéraux satisfont φ_j, on en choisit arbitrairement un pour ne pas visiter plusieurs fois le sommet) et donc de parcourir tous les sommets du graphe une et une seule fois en cycle hamiltonien.

Vice-versa, on interprète un cycle hamiltonien dans le graphe comme affectation des variables de φ en fonction de la direction de parcours de chaque gadget correspondant à une variable. Comme le cycle visite chaque sommet, en particulier ceux qui correspondent aux clauses de la formule, cela nous garantit que chaque clause, et donc la formule elle-même, est satisfaite par cette affectation.

On peut conclure que la transformation d'une formule en graphe proposée est bien une réduction many-one polynomiale, ce qui implique que le problème **Cycle Hamiltonien** est NP-complet.