TD 09 – Révisions – Correction des exercices 1, 2 et 3

Exercice 1. *sur la taille des entrées et des entiers*

✎ On peut remarquer que si un entier x n'est pas premier, alors il a forcément un diviseur au plus égal à $\sqrt{x}$.

```
pour y de 1 à ⌊√x⌋ faire
   si x modulo y == 0 alors
     rejeter
accepter
```

Soit n la taille de l'entrée, donc $n = \log_2(x)$, c'est-à-dire $x = 2^n$. Il y a $\sqrt{x} = x^{\frac{1}{2}} = 2^{\frac{n}{2}}$ passages dans la boucle, et chaque passage dans la boucle prend un temps constant pour le calcul du modulo (en considérant que c'est une instruction élémentaire). $2^{\frac{n}{2}}$.

Exercice 2. *P et la complémentation*

✎ L'idée consiste à inverser la réponse de l'algorithme qui résout A, pour obtenir un algorithme de même complexité qui résout co-A.

Soit $A \in \mathsf{P}$ et M une machine de Turing déterministe qui décide le langage A en temps polynomial. Soit M' la machine de Turing déterministe identique à A, excepté pour les états d'acceptation et de rejet, qui sont inversés : les transitions qui mènent à q_a dans M sont transformées en des transitions qui mènent à q_r dans M', et les transitions qui mènent à q_r dans M sont transformées en des transitions qui mènent à q_a dans M'. Alors M' accepte si et seulement si M rejette, donc M' décide bien co-A. De plus M' fonctionne en un temps identique à M, c'est-à-dire polynomial. Donc co-$A \in \mathsf{P}$

Remarque : attention, la dernière phrase ne serait pas vraie si nous considérions des machines M (et donc aussi M') non-déterministes, puisque le critère d'acceptation/rejet de ces dernières est : acceptation si et seulement si il existe une *branche d'exécution* qui accepte.

Exercice 3. **Noyau $\leq_m^p$ SAT**

2. On donne un algorithme non-déterministe polynomial pour résoudre le problème **Noyau**.

```
Résoudre_Noyau(G = (V, A))
   deviner(N ⊆ V)
   pour chaque couple de sommets x, y ∈ N faire
     si (x, y) ∈ A alors
       rejetter
   pour chaque sommet x ∉ N faire
     trouve_y=faux
     pour chaque y ∈ N faire
       si (x, y) ∈ A alors
         trouve_y=vrai
     si trouve_y=faux alors
       rejetter
   accepter
```

Sur le temps. On devine bien des objets de taille polynomial (un sous-ensemble N de V peut être vu comme une suite de $|V|$ bits qui indique (à 1) les sommets de V qui sont dans N, et ceux qui ne le sont pas (à 0)). De plus, chaque *branche d'exécution* fonctionne en temps polynomial : deux boucle en au plus $|V|^2$ étapes chacune.

Sur la correction. La première boucle vérifie la première condition de la définition du noyau et rejette si elle n'est pas respectée, la seconde boucle vérifie la seconde condition de la définition du noyau et rejette si elle n'est pas respectée. Ainsi une branche qui accepte doit correspondre à un $N \subset V$ qui est un noyau, donc l'algorithme accepte si et seulement si il existe un noyau, c'est-à-dire qu'il décide bien le problème **Noyau**.

Donc **Noyau** $\in \mathsf{NP}$.

3. Soit $G = (V, A)$ un graphe à n sommets. Pour chaque sommet $i \in V$ on crée une variable propositionnelle p_i (qui a vocation a être vraie lorsque i est dans le noyau de G), puis on considère les formules suivantes :

$$\phi_1 = \bigwedge_{(i,j)\in A} (p_i \to \neg p_j) \qquad \phi_2 = \bigwedge_{i\in V} \left(p_i \vee \bigvee_{(i,j)\in A} p_j \right) \qquad \phi_G = \phi_1 \wedge \phi_2.$$

Montrons que G admet un noyau $\iff$ ϕ_G est satisfaisable.

$\boxed{\Leftarrow}$ Soit v un modèle de ϕ_G (une valuation qui satisfait ϕ). On pose $N = \{i \in V \mid v(p_i) = 1\}$. Alors, d'une part, pour tout arc $(i, j) \in A$, i ou j n'est pas dans N (puisque v satisfait ϕ_1), ce qui signifie que les sommets de N sont deux à deux non-adjacents. D'autre part, pour tout sommet $i \notin N$ on a $v(p_i) = 0$ et donc, puisque v satisfait ϕ_2, $v(\bigvee_{(i,j)\in A} p_j) = 1$, ce qui impose que l'une des arêtes partant de i arrive sur un sommet $j \in N$. Ainsi, N est bien un noyau de G.

$\boxed{\Rightarrow}$ Soit N un noyau de G. Considérons la valuation v sur les variables de ϕ_G définie par : $v(p_i) = 1$ ssi $i \in N$. Puisque aucun arc de G n'a ses deux extrémités dans N, on a $i \in N \implies j \notin N$ pour tout $(i, j) \in A$. Et donc, par construction de v : $v(p_i \to \neg p_j) = 1$ pour tout $(i, j) \in A$, d'où s'ensuit que v satisfait ϕ_1. De plus, un sommet $i \in V$ est soit dans N, soit relié à un sommet de N, ce qui signifie que l'un des arcs partant de i arrive dans N. Ceci induit l'égalité $v(p_i \vee \bigvee_{(i,j)\in A} p_j) = 1$ pour tout $i \in V$. La satisfaction de ϕ_2 par v s'en déduit. Finalement, on a montré que v satisfait $\phi_1 \wedge \phi_2 = \phi_G$, preuve que cette dernière formule est satisfaisable.

4. Nous voulons montrer que **SAT** $\leq_m^p$ **Noyau**, or **SAT** est NP-complet et **Noyau** $\in \mathsf{NP}$, donc on pourra en déduire que **Noyau** est NP-complet.

Nous présenterons seulement l'idée de la réduction. Soit ϕ une formule, nous construisons le graphe suivant G_ϕ suivant.

— Pour chaque clause C_i de ϕ, nous créons trois sommets C_i^1, C_i^2, C_i^3 et un triangle $C_i^1 \to C_i^2 \to C_i^3 \to C_i^1$. **Idée :** ces triangles vont empêcher que l'on essaye de mettre ces sommets dans le noyau. Il devront avoir un voisin sortant dans le noyau…

— Pour chaque variable x_j de ϕ, nous créons deux sommets $x_j, \neg x_j$ et un digone $x_j \to \neg x_j \to x_j$. **Idée :** on aura dans le noyau exactement l'un de ces deux sommets, ce qui correspondra à une valuation (les sommets du noyau seront évalués à $\top$).

— Pour chaque littéral ℓ_j de chaque clause C_i, nous ajoutons les trois arcs : $C_i^1 \to \ell_j$, $C_i^2 \to \ell_j$ et $C_i^3 \to \ell_j$. **Idée :** ces arcs permettrons au littéral qui satisfait la clause d'être le voisin sortant des sommets correspondant à la clause, comme indiqué au premier point.