Tutorial 5 : Inputs encoding

Computational complexity theory, 5th semester.

2022

Exercice 1 — *Weakly and strongly NP-Complete*

For each of the NP-Complete problems of the first tutorial, determine if it is strongly NP-Complete or weakly NP-Complete.

▶ **Correction**

(HAM) : les instances sont des graphes G de taille n. On a donc $|x| = n$, $l(x) = n$, $\max(x) = 0$. Ce problème est NP-Complet au sens fort car il n'y a pas d'entier. (On considère que $|x| = n$ mais en vrai, cette taille vaut au moins m, le nombre d'arêtes puisqu'il faut encoder chaque arête. Puisque le nombre d'arêtes est inférieure à n^2, dans le cadre de notre cours ça n'a pas d'importance.)

(3SAT) : les instances sont des formules booléennes sur n variables avec m clauses. On a donc $|x| = n + 3m$, $l(x) = n + 3m$ et $\max(x) = 0$. Ce problème est NP-Complet au sens fort car il n'y a pas d'entier.

(VERTEX COVER) *Soit G un graphe non orienté et K un entier, existe-t-il un sous-ensemble des nœuds de G de taille inférieur à K et couvrant toutes ses arêtes.* Les instances sont des graphes G de taille n et un entier $K \in \mathbb{N}$. On a donc $|x| = n + \log(K)$, $l(x) = n + 1$, $\max(x) = K$. On peut voir que ce problème est trivial si $K > n$ puisqu'on peut alors prendre tous les nœud. On peut donc logiquement penser qu'une réduction polynomiale de Karp de 3SAT vers Vertex Cover se restreindra au cas où $K \leq n$, donc au cas où $\max(x) \leq l(x)$. La réduction est là : `http://web.mit.edu/~neboat/www/6.046-fa09/rec8.pdf`. Ce problème est donc NP-Complet au sens fort.

(INS) *Soit G un graphe non orienté et K un entier, existe-t-il un sous-ensemble des nœuds de G de taille inférieur à K et non reliés deux à deux.* Les instances sont des graphes G de taille n et un entier $K \in \mathbb{N}$. On a donc $|x| = n + \log(K)$, $l(x) = n + 1$, $\max(x) = K$. On peut voir que ce problème est trivial si $K > n$ puisqe la réponse est nécessairement non. On peut donc logiquement penser qu'une réduction polynomiale de Karp de 3SAT vers INS se restreindra au cas où $K \leq n$, donc au cas où $\max(x) \leq l(x)$. Une idée de réduction est là : `https://stackoverflow.com/questions/47657574/how-can-3-sat-be-reduced-to-independent-set`. Ce problème est donc NP-Complet au sens fort.

(SET COVER) *Soit X un ensemble et S des sous-ensembles de X et K un entier, existe-t-il un sous-ensemble C de S de taille inférieure à K et couvrant X.* Les instances sont des ensembles X de taille n, p sous-ensembles de X et un entier $K \in \mathbb{N}$. On a donc $|x| = n \cdot + \log(K)$, $l(x) = n \cdot p + 1$, $\max(x) = K$. On peut voir que ce problème est trivial si $K > p$ ou $n = 0$ puisqe la réponse est nécessairement oui. On peut donc logiquement penser qu'une réduction polynomiale de Karp de (3SAT) vers (SET COVER) se restreindra au cas où $K \leq p$ et $n \geq 1$, donc au cas où $\max(x) \leq l(x)$. La réduction dans le TD3 vérifie cette propriété. Ce problème est donc NP-Complet au sens fort.

(W-VERTEX COVER) *Soit G un graphe non orienté, des poids $\omega : V \to \mathbb{N}$ et K un entier, existe-t-il un sous-ensemble des nœuds de G de poids total inférieur à K et couvrant toutes ses arêtes.* Les instances sont des graphes G de taille n, des poids ω et un entier $K \in \mathbb{N}$. On a donc $|x| = n + \sum_{v \in V} \log(\omega(v)) + \log(K)$, $l(x) = 2n + 1$, $\max(x) = \max_{v \in V}(\log(\omega(v)), K)$. On peut voir que

ce problème reste NP-Complet si on se restreint au cas où $\omega(v) = 1$ pour tout v, puisqu'on a alors le problème (VERTEX COVER). Considérons la réduction de (VERTEX COVER) vers (W-VERTEX COVER) qui, connaissant une instance x de (VERTEX COVER) fixe tous les poids à 1 pour former une instance y de (W-VERTEX COVER). On a alors $\max(y) = K$ qui est borné polynomialement par rapport à $\max(x)$ et $l(x)$. Puisque (VERTEX COVER) est NP-Complet au sens fort, on peut appliquer le dernier théorème du cours pour prouver que (W-VERTEX COVER) est NP-Complet au sens fort.

De même on peut prouver que (WINS) et (W-SET COVER) sont NP Complets au sens fort.

(LOP) : soit un graphe orienté $G = (V, A)$ de taille n, des poids sur les arêtes $\omega(a)$, un noeud s, une noeud t, et un entier K, existe-t-il un chemin élémentaire de s vers t de poids supérieur à K ? On a donc $|x| = n + \sum_{a \in A} (\log(\omega(a))) + 2 + \log(K)$, $l(x) = n + |A| + 2 + 1$ et $\max(x) = \max_{a \in A}(\omega(a), K)$. Il existe une réduction de (HAM) vers (LOP) qui consiste à garder le graphe G de l'instance de (HAM) de mettre des poids de 1 sur toutes les arêtes et poser $K = n - 1$, rajouter un noeud s, et un noeud t, relier s à tous les nœuds sauf t avec des arcs de poids 0, relier tous les nœuds sauf s à t avec des arcs de poids 0. Sur ces instances, $\max(x) = n - 1 \leq l(x)$, donc (LOP) est NP Complet au sens fort.

(SUBSET SUM) : un ensemble X de k entiers et un entier B. Existe–til un sous-ensemble de X dont la somme fait B. On a donc $|x| = \log(B) + \sum_{x \in X} (\log(x))$, $l(x) = k + 1$, $\max(x) = \max(B, x)$. Il existe un algorithme en $O(kB)$ pour résoudre (SUBSET SUM) (cf Chapitre 1 du cours de RO). Cet algo est pseudo polynomial car $k \leq k + 1 = l(x)$ et $B \leq max(B, xi) = \max(x)$ donc $O(kB) = O(l(x)max(x))$ donc polynomial en $l(x)$ et $max(x)$.

Or (SUBSET SUM) est NP Complet donc il est NP Complet au sens faible.

Alternative : si B est codé en unaire, $O(kB)$ est polynomial, donc SUBSET SUM est NP Complet mais polynomial si on code ses entiers en unaire. Donc NP Complet au sens faible.

(PARTITION) : un ensemble X de k entiers. Existe-t-il une partition de X dont la somme des deux parties est égale. On a donc $|x| = \sum_{x \in X} (\log(x))$, $l(x) = k$, $\max(x) = \max(X)$. Il existe un algorithme en $O(k \sum \lim_{x \in X} X)$ pour résoudre (PARTITION) (Cet algo est le même que pour SUBSET SUM). Cet algo est pseudo polynomial car $k \leq k + 1 = l(x)$ et $\sum \lim_{x \in X} \leq k \cdot max(X) = l(x) \max(x)$ donc $O(kB) = O(l(x)^2 max(x))$ donc polynomial en $l(x)$ et $max(x)$.

Or (PARTITION) est NP Complet donc il est NP Complet au sens faible.

Alternative : si X est codé en unaire, $O(k \sum \lim_{x \in X} X)$ est polynomial, donc (PARTITION) est NP Complet mais polynomial si on code ses entiers en unaire. Donc NP Complet au sens faible.

Une preuve similaire montre que (KNAPSACK) est NP-Complet au sens faible.

Exercice 2 — *Some results*

1. What prove a reduction from a weakly NP-Complete problem ?

▶ **Correction**

Si Π_1 est faiblement NP-Complet et $\Pi_1 \preccurlyeq \Pi_2$, alors on sait que Π_2 est NP-Difficile. Cela ne prouve pas que Π_2 est fortement ou faiblement NP-Complet. (Rappelez vous qu'on peut réduire tous les problèmes de NP à 3SAT qui est fortement NP-Complet.)

Dans tous les cas, il n'est pas possible d'appliquer le derner théorème du cours.

2. Show that a polynomial Karp reduction from a strongly NP-Complete problem Π_1 to a problem Π_2 does not prove that Π_2 is strongly NP-Complete.

▶ **Correction**

Il existe une réduction de (3SAT) vers (SUBSET SUM) et ce dernier est NP-Complet au sens faible. Pour que la NP Complétude soit au sense fort, il faut en plus que l'entier $max(x)$ de l'instance de (SUBSET SUM) produite soit borné polynomialement par rapport à la taille de l'instance de (3SAT).

3. Prove that, if a problem is strongly NP-Complete (which means that it is NP-Complete if its integers are unary encoded), the problem is still NP-Complete if its integers are binary encoded.

▶ **Correction**

Considérons les deux versions Π_u et Π_b du problèmes où les entiers sont respectivement codés en unaire et en binaire.

Il existe une réduction polynomiale de Karp de Π_u vers Π_b qui consiste à réencoder tous les entiers en binaire. La réduction préserve trivialement les instances positives et négatives. Elle est polynomiale (même meilleure que polynomiale) car il faut un temps $O(\log(n))$ pour encoder un entier n en binaire. Puisque , dans Π_u, l'instance est encodée en unaire, la taille de l'entier n est n, on a bien une transformation en temps polynomial.

Exercice 3 — *Weakly and strongly Polynomial*

1. Is the simplex algorithm solving linear programs strongly polynomial ? We now restrict the instances to programs with at most 2 variables, is the simplex algorithm, in that case, strongly polynomial ?

▶ **Correction**

Un programme linéaire est constitué d'une matrice A, d'un vecteur b et d'un vecteur c et on cherche le vecteur x maximisant cx tel que $Ax \leq b$. L'algorithme du simplexe se déplace de solution de base réalisable en solution de base réalisable jusqu'à trouver une solution optimale.

On note n le nombre de variables (et colonnes de A) et m le nombre de contraintes (et lignes de A).

Aujourd'hui, on ne sais pas si l'algorithme du simple est polynomial, car on ne connait pas d'algorithme polynomial permettant d'éviter de cycler entre des solutions de bases réalisables. On ne sait donc pas s'il est fortement polynomial. Par contre, on peut voir que chaque itération de l'algorithme du simplexe est linéaire en le nombre de variables. Si on ne cycle pas entre les solutions de bases, on ne peut pas avoir plus d'itérations que le nombre de solutions de bases. Il y a au plus m^n solutions de bases réalisables dans un programme linéaire. Enfin, chaque itération effectue $O(n)$ opérations élémentaires ou arithmétiques.

Donc dans le cas de $n = 2$ variables, en supposant que le simplexe ne boucle pas, il s'exécute en $O(m^2)$ et est donc fortement polynomial. C'est aussi le cas pour n'importe quel nombre fixé de variables.

2. Describe a strongly polynomial algorithm to compute the mean of n numbers.

▶ **Correction**

L'algorithme naif suivant est fortement polynomial.

Entrées: n nombres $x_1, x_2, \ldots, x_n$
Sorties: La moyenne de ces nombres

1: $s \leftarrow 0$
2: **Pour** $i \leq n$ **Faire**
3: $\quad s \leftarrow s + x_i$
4: **Renvoyer** s/n.

En considérant les opérations arithmétiques comme élémentaires, la complexité de cet algorithme est $O(n)$, donc polynomial en $l(x)$.

3. We want to compute the n-th digit of 2 with the Newton and Raphson method.
   — Let f be a strictly convex and increasing function on $D \subset \mathbb{R}$, we search for a real $x \in D$ such that $f(x) = 0$. (We assume that such a real exists). Let $x_0 \in D$ and $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$, show that the sequence $(x_n)_{n\in\mathbb{N}}$ is convergent and approaches x.
   — Show that $x_n - x \leq f(x_n)/f'(x)$
   — Let $f(x) = x^2 - 2$. Describe an algorithm to compute the n-th digit of $\sqrt{2}$.
   — Show that, if $x_0 < \sqrt{2} + 0.3$,the algorithm stops in n iterations. Deduce the complexity of the algorithm. Is it pseudo polynomial, strongly polynomial or weakly polynomial ?

   ▶ **Correction**

   Pour cette dernière question, la réponse dépend de l'encodage de n. Si n est binaire, cet algorithme est exponentiel. S'il est unaire, alors l'algorithme est faiblement polynomial puisque la complexité dépend de $\max(x)$ et n'est donc pas borné uniquement par rapport à $l(x)$. Par contre il est pseudo-polynomial.

Exercice 4 — *Succinct graph* A boolean circuit C is a directed graph with no circuit where the n sources are the inputs (1 or 0), the sink is unique and is an output (1 or 0 too) and the nodes are logical operators AND, OR and NOT. For an input $x \in 0, 1^n$, we write $C(x)$ the associated output.

Let G be a undirected graph with 2^n nodes numbered from 1 to 2^n. We define the succinct representation of G as a boolean circuit C with $2n$ inputs (in other words two binary numbers x and y of size n) such that, for every couple of nodes number x and y, the output $C(x, y) = 1$ if and only if x and y are linked by an edge. Warning : not all the graphs have a succinct representation.

We want to show that the following problem is NP-Complete.

(TRIANGLE-SUCC ?) : let C be a boolean circuit representing a graph G, does G have a triangle (a clique of size 3) ?

1. Show that (TRIANGLE-SUCC ?) belongs to NP.
2. Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ such that $V_1 \subset V_2$ and such that those two graphs have a succinct representation, show that $G = (V_2, E_1 \cup E_2)$ has a succinct representation.
3. Let φ be an instance of (SAT ?) with n variables $x_1, x_2, \ldots, x_n$. Let $G_\varphi = (V_\varphi, E_\varphi)$ the following graph. V_φ contains $2^n + 1$ nodes : $(v_0, v_1, \ldots, v_{2^n})$. We link v_i to v_{2^n} in E_φ if and only if, when TRUE is affected to x_j if the j-th bit of the binary representation of i is 1 and FALSE otherwise, φ is satisfied. Show that G_φ has a succinct representation.
4. Let $G_2 = (V_2, E_2)$ be the graph containing $2^n + 2$ nodes $(v_0, v_1, \ldots, v_{2^n+1})$ and where every node is linked to v_{2^n+1} (except v_{2^n+1} itself). Show that G_2 has a succinct representation.
5. Deduce that $G = (V_2, E_\varphi \cup E_2)$ has a succinct representation.
6. Show that G has a triangle if and only if φ can be satisfied.
7. Deduce that (TRIANGLE-SUCC) is NP-Complete.