TD 04 – Problèmes NP-complets

Exercice 1. *Cycle hamiltonien et voyageur de commerce*

Dans un graphe non-orienté sans boucle $G = (V, E)$, un *cycle hamiltonien* est un cycle qui passe exactement une fois par chaque sommet. Le problème de décision **Hamiltonien** consiste, étant donné un graphe G, à décider s'il possède un cycle Hamiltonien.

1. Donner un exemple de graphe connexe qui ne possède pas de cycle hamiltonien.
2. Montrer que le problème **Hamiltonien** appartient à la classe NP.

Le problème du **Voyageur de commerce** (*Traveling-salesman problem*, **TSP**) consiste à optimiser la tournée d'un agent qui doit passer par un ensemble de villes. Formellement, étant donné un graphe complet $G = (V, E)$, une fonction de coût $c : V \times V \to \mathbb{Z}$ et un entier $k \in \mathbb{Z}$, on souhaite décider s'il existe un cycle hamiltonien dont la somme des coûts des arêtes est k ou moins.

3. Pourquoi le graphe G est-il pris complet ?
4. Montrer que le problème **TSP** appartient à la classe NP.

Le problème **Hamiltonien** est NP-complet, et nous allons le réduire à **TSP**.

5. Expliquer comment, si vous aviez accès à un algorithme efficace de résolution du problème **TSP**, vous pourriez l'utiliser pour résoudre le problème **Hamiltonien**.
6. Que peut-on en déduire sur la difficulté du problème **TSP** ?

Exercice 2. *Stable maximum*

Dans un graphe non-orienté $G = (V, E)$ (sans boucle), un *stable* est un sous-ensemble de sommets qui n'ont aucune arête entre eux, c'est-à-dire $X \subset V$ tel que $E \cap V^2 = \emptyset$. Un stable X est *maximum* lorsqu'il n'existe pas de stable Y strictement plus grand (avec $|X| < |Y|$). Le problème de décision **Stable** consiste, étant donné un graphe G et un entier k, à décider si G possède un stable de taille au moins k.

1. Pourquoi **Stable** est-il formulé avec « au moins k » et pas « exactement k » ?

Supposons que nous ayons accès à un programme efficace pour résoudre le problème **Stable**, appelé avec `stable(G,k)`, dont on ne connaît pas l'implémentation (c'est une « boîte noire »).

2. Expliquer comment utiliser cette boîte noire pour retourner un stable maximum. Évaluer sa complexité, en considérant que l'appel à `stable` est en temps constant.
3. Donner un algorithme efficace de résolution du problème **Stable** pour les graphes non-orientés sans boucle où tous les sommets sont de degré exactement 2.

Exercice 3. Subset-sum *et* Partition

Le problème **Subset-sum** consiste, étant donnée une liste d'entiers naturels $e_1, \ldots, e_n$ et un objectif s, à décider s'il existe une sélection des entiers dont la somme est s.

1. Montrer que le problème **Subset-sum** appartient à la classe NP.

Le problème **Partition** consiste, étant donnée une liste d'entiers naturels $e_1, \ldots, e_n$, à décider s'il existe une parition des entiers en deux parties dont les sommes sont égales.

2. Montrer que le problème **Partition** appartient à la classe NP.

Ces deux problèmes sont NP-complets, et nous allons les réduire l'un à l'autre.

3. Expliquer comment, si vous aviez accès à un algorithme efficace de résolution du problème **Subset-sum**, vous pourriez l'utiliser pour résoudre le problème **Partition**.

4. Expliquer comment, si vous aviez accès à un algorithme efficace de résolution du problème **Partition**, vous pourriez l'utiliser pour résoudre le problème **Subset-sum**.