Université Lille1 Sciences et Technologies

UFR IEEA

AAC- TP

2012–2013
Master 1 d'Informatique

TP 2 - Les propriétés NP

Préambule: Vous trouverez sur le portail une archive avec des sources et des jeux de données. Deux séances encadrées sont prévues sur ce sujet. Le TP sera à rendre sous Prof, la date limite pour la remise du TP étant fixée par votre chargé de TP. Les réponses aux questions qui ne nécessitent pas d'implémentation sont à inclure dans le compte-rendu de TP qui peut être en format txt ou pdf.

Objectif: Le but du TP est de concrétiser les notions de propriétés NP et de réductions polynomiales vues en cours. Le problème central que l'on va étudier est TSP, le problème du voyageur de commerce (Travelling Salesman Problem) dont le but est de trouver une tournée de villes à visiter la plus courte possible. Le problème de décision est, étant données les distances entre n villes et une longueur maximale de tournée, de proposer une tournée qui visite les villes une et une seule fois, revient au point de départ, dont la longueur soit inférieure ou égale à la longueur donnée. Formellement, on a donc:

Travelling Salesman Problem

Donnée

n, un entier – un nombre de villes

D, une matrice (n, n) d'entiers – elle représente les distances, qu'on suppose entières

l, un entier – la longueur maximale "autorisée", entière

Sortie

Oui, s'il existe une tournée possible, i.e.

$tour : [0 \dots n-1] \to [0 \dots n-1]$ – une **permutation** des n villes

telle que:

$D(tour[n-1], tour[0]) + \sum_{i=0}^{n-2} D(tour[i], tour[i+1])_i \leq l$,

Non, sinon.

Exemple: soit $n = 4$ et D donnée par les distances suivantes:

	A	B	C	D
A	0	2	5	7
B	7	0	8	1
C	2	1	0	9
D	2	2	8	0

La longueur minimale d'une tournée est 9 avec la tournée A, C, B, D, A. Donc, le problème aura une solution pour $l = 9$, il n'en aura pas pour $l = 8$.

Le problème TSP est très connu et très étudié, voir par exemple le site: http://www.tsp.gatech.edu/.[1]
Il y a de nombreuses versions du problème, par exemple:
. symétrique ou non, selon que l'on suppose que la distance entre i et j est la même que celle entre j et i.
. on peut supposer que les distances vérifient l'inégalité triangulaire: $d(i, j) \leq d(i, k) + d(k, j)$.
On se placera ici a priori dans le cas général (non nécessairement symétrique sans condition sur la distance).

[1] Un film Traveling Salesman est sorti cette année, l'intrigue étant basée sur la résolution de ce problème.

La classe NP

On définira plusieurs classes abstraites correspondant aux différentes classes de problèmes de décision. La classe `PbDecision` correspond juste à la classe générale des problèmes de décision ou propriétés:

```
public abstract class PbDecision {
    public abstract boolean aUneSolution();
}
```

La classe abstraite `ExpTime` étend la classe `PbDecision` en rajoutant juste sous forme de commentaire une contrainte sur la complexité de l'algorithme de décision:

```
public abstract class ExpTime extends PbDecision{
    //doit être de complexité temporelle au plus exponentielle
    public abstract boolean aUneSolution();
}
```

La classe abstraite `NP` utilise l'interface `Certificat` et correspond à la classe des propriétés NP définie par les certificats:

```
public abstract class NP extends ExpTime{

    //on doit pouvoir définir pour le pb un certificat (une notion plutot qu'un objet...)
    abstract public Certificat cert();

    abstract public boolean estCorrect(Certificat cert);
.....
}
```

Important: Une architecture logicielle est proposée mais vous pouvez bien sûr l'améliorer. Les sources proposées sont disponibles dans l'archive. Il y a des exemples de programmes de test: attention, ils sont compilables mais leur exécution provoquera une erreur tant que vous n'aurez pas écrit certaines méthodes. Vous avez quelques fichiers de données dans l'archive de `donnees`. Il existe de nombreux exemples de données (par exemple http://www.tsp.gatech.edu/data/index.html). Attention à ne pas tester sur des données de grande taille pour la recherche exhaustive.

Q 1. Pourquoi peut-on dire que `NP` étend `ExpTime`?
Q 2. *TSP est NP : la notion formelle de certificat et d'algorithme de vérification*
Q 2.1. Définir une notion de certificat pour TSP. Quelle est la taille d'un certificat par rapport à la taille du problème?
Q 2.2. Proposer un algorithme de vérification d'un certificat.
Q 2.3. En déduire que TSP est bien NP.
Q 2.4. Pour n donné, combien de valeurs peut prendre un certificat?

Q 3.[à coder] *TSP est NP: L'implémentation*
Ecrire la classe `Certificat_TSP` qui implémente l'interface `Certificat` et la classe `TSP` qui étend `NP`: pour cette question, pour `Certificat` n'écrire que le constructeur et les méthodes `saisie()` et `display()`.
Vous pouvez tester la correction avec le mode "-verif" du programme de test.

Q 4.[à coder] *NP= non-déterministe polynomial*
Ecrire la méthode `alea()` de `Certificat_TSP`. Vous pouvez tester le mode "-nondet".

Q 5.[à coder] *$NP \subset ExpTime$.*
Pour pouvoir déduire un algorithme de décision exponentiel, on doit pouvoir explorer tous les certificats.
Pour cela implémenter les méthodes `reset()`, `suivant()`, `estDernier()`, basées sur un ordre total des valeurs possibles des certificats -à définir-:

```
//affecte au certificat la première valeur pour l'ordre choisi
public void reset();

//retourne vrai si la valeur est la dernière dans l'ordre choisi, faux sinon
public boolean estDernier();

//modifie la valeur du certificat en la suivante pour l'ordre
//le comportement n'est pas défini si le certificat est le dernier
public void suivant();
```

Vous pouvez tester le mode "-exhaust", sur des données de petite taille.

Les Réductions Polynomiales

On va maintenant implémenter des réductions entre problèmes. On va étudier les problèmes de cycle et chemin hamiltoniens dans un graphe. On représentera ici les graphes sous forme de leur matrice d'adjacence:

Hamilton Cycle

Donnée

n, un entier – un nombre de sommets

D, une matrice (n, n) de booléens – elle représente les arêtes

Sortie

Oui, s'il existe un cycle hamiltonien, i.e.

$ham : [0..n-1] \to [0..n-1]$ – une **permutation** des n villes

telle que:

$D[ham[n-1]][ham[0]] = true$

$D[ham[i]][ham[i+1]] = true$, pour tout $0 \leq i \leq n-2$

Non, sinon.

Hamilton Path

Donnée

n, un entier – un nombre de sommets

D, une matrice (n, n) de booléens – elle représente les arêtes

Sortie

Oui, si il existe un chemin hamiltonien, i.e.

$ham : [0..n-1] \to [0..n-1]$ – une **permutation** des n villes

telle que:

$D[ham[i]][ham[i+1]] = true$, pour tout $0 \leq i \leq n-2$

Non, sinon.

Ces deux propriétés sont connues $NP-$complètes.

Q 6. $HamiltonCycle \leq_P TSP$

Q 6.1. Montrer que $HamiltonCycle$ se réduit polynomialement dans TSP.

Q 6.2.[à coder] Implémenter la réduction polynomiale de $HamiltonCycle$ dans TSP en utilisant la classe abstraite `NPRed`. Vous pouvez tester avec le programme de test et les données fournis.

Q 6.3. Qu'en déduire pour TSP?

Q 6.4. Pensez-vous que TSP se réduise polynomialement dans $HamiltonCycle$? Pourquoi?

Q 7. $HamiltonPath \leq_P HamiltonCycle$
Q 7.1. Montrer que $HamiltonPath$ se réduit dans $HamiltonCycle$.
Q 7.2. [à coder] Implémenter une réduction polynomiale $HamiltonPath$ dans $HamiltonCycle$ en utilisant la classe abstraite `NPRed`. Vous pouvez tester avec le programme de test et les données fournis.
Q 7.3. Montrer que $HamiltonPath$ se réduit dans TSP.

Propriétés versus Problèmes d'Optimisation

Au problème de décision TSP, on peut associer deux problèmes d'optimisation:

TSPOpt1

Donnée

n, un entier – un nombre de villes

D, une matrice (n, n) d'entiers – elle représente les distances, qu'on suppose entières

Sortie

l, minimale telle qu'il existe une tournée possible, i.e.

$tour : [0 \dots n-1] \to [0 \dots n-1]$ – une **permutation** des n villes

telle que:

$D(tour[n-1], tour[0]) + \sum_{i=0}^{n-2} D(tour[i], tour[i+1])_i = l$,

TSPOpt2

Donnée

n, un entier – un nombre de villes

D, une matrice (n, n) d'entiers – elle représente les distances, qu'on suppose entières

Sortie

Une tournée de longueur minimale, i.e.

$tour : [0 \dots n-1] \to [0 \dots n-1]$ – une **permutation** des n villes

telle que:

$D(tour[n-1], tour[0]) + \sum_{i=0}^{n-2} D(tour[i], tour[i+1])_i$ soit minimale dans les tournées possibles.

Q 8. Montrer que si $TSPOpt1$ (resp. $TSPOpt2$) était P, la propriété TSP le serait aussi ; qu'en déduire pour $TSPOpt1$ (resp. $TSPOpt2$)?
Q 9. Montrer que si la propriété TSP était P, $TSPOpt1$ le serait aussi.
Q 10. *Plus dur...* Montrer que si la propriété TSP était P, $TSPOpt2$ le serait aussi.