Université de Lille 1 SCIENCES ET TECHNOLOGIES

UFR IEEA

ACT- TP

2015–2016
Master 1 d'Informatique

TP - La classe NP

Objectif: Le but du TP est de "concrétiser" les notions de propriété NP et de réduction polynomiale vues en cours.

A faire: Le TP a trois sections, une sur la notion de NP, un sur celle de réduction polynomiale, une dernière (sans implémentation) sur le lien optimisation/décision. Le TP est à rendre la semaine du 16 novembre sous Prof sous forme d'une archive contenant un rapport (bref) avec les réponses aux questions et vos choix d'implémentation et le code. Vous disposez d'exemples de données pour tester.

1 Qu'est-ce qu'une propriété NP?

On va illustrer la notion de propriété NP via le problème de TSP, le problème du voyageur de commerce (Traveling Salesman Problem) dont le but est de trouver une tournée de villes à visiter la plus courte possible. Le problème de décision est donc: étant données les distances entre n villes et une longueur maximale de tournée, existe-t-il 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 de longueur au plus l, i.e. $tour : \{0 \dots n-1\} \to \{0 \dots n-1\}$, une **permutation** des n villes telle que $D[tour(n-1), tour(0)] + \sum_{i=0}^{n-2} D[tour(i), tour(i+1)]$ soit inférieure ou égale à 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 $l = 9$ (ou 10 ou toute valeur supérieure), il n'en aura pas pour $l = 8$.

Le problème TSP est très connu, voir par exemple le site: http://www.math.uwaterloo.ca/tsp/. [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, ou supposant ou non 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 dont l'intrigue est basée sur la résolution de ce problème est sorti en 2012.

A faire avant d'implémenter!

Q 1. *La propriété est NP.*

NP

L est dit NP si il existe un polynôme Q, et un algorithme polynomial A à deux entrées et à valeurs booléennes tels que:

$$L = \{u / \exists c, A(c,u) = Vrai, |c| \leq Q(|u|)\}$$

Définir une notion de certificat.

Comment pensez-vous l'implémenter?

Quelle sera la taille d'un certificat? La taille des certificats est-elle bien bornée polynomialement par rapport à la taille de l'entrée?

Proposez un algorithme de vérification associé. Est-il bien polynomial?

Q 2. *NP = Non déterministe polynomial*

Q 2.1. *Génération aléatoire d'un certificat.*

Proposez un algorithme de génération aléatoire de certificat, i.e. qui prend en entrée une instance de TSP, ou seulement le nombre de villes, et génère aléatoirement un certificat de façon à ce que chaque certificat ait une probabilité non nulle d'apparaître.

Votre algorithme génère-t-il de façon uniforme les certificats, i;e. tous les certificats ont-ils la même probabilité d'apparaître?

Q 2.2. Quel serait le schéma d'un algorithme non-déterministe polynomial pour le problème?

Q 3. *$NP \subset ExpTime$*

Q 3.1. Pour n fixé, combien de valeurs peut-prendre un certificat?

Q 3.2. *Enumération de tous les certificats*

Une méthode classique pour énumérer les certificats (soit définir un itérateur sur les certificats) consiste à s'appuyer sur un ordre total sur les certificats associés à un problème. Donc, on définit le certificat de départ "le plus petit" pour l'ordre, et la notion de successeur qui permet de passer d'un certificat au suivant.

Quel ordre proposez-vous?

Q 3.3. *L'algorithme du British Museum*

Comment déduire de ce qui précède un algorithme pour tester si le problème a une solution? Quelle complexité a cet algorithme?

Implémentation

On veut donc implémenter les notions et algorithmes évoqués ci-dessus. On devra donc être capable de lire une instance du problème TSP, lire une proposition de certificat, vérifier si un certificat est valide, vérifier si le problème a une solution en essayant tous les certificats, "vérifier aléatoirement" si le problème a une solution en générant aléatoirement un certificat.

Sur le portail (cf fin du sujet) sont proposés un embryon d'architecture et programme de test en Java. **Vous pouvez en choisir une autre ou utiliser d'autres langages (C, CAML, HASKELL, ..)**. Vous veillerez à documenter votre code, à respecter l'esprit et à faciliter le test en respectant le schéma ci-dessous.

Pour tester, on pourra donc avoir un programme qui lit l'instance du problème dans un fichier et :

. en mode "vérification" propose à l'utilisateur de saisir un certificat et vérifie sa validité.

. en mode "non-déterministe", génère aléatoirement un certificat, le teste et retourne Faux si il n'est pas valide, "Vrai" sinon (avec éventuellement la valeur du certificat).

. en mode "exploration exhaustive" génère tous les certificats jusqu'à en trouver un valide, si il en existe un et retourne Faux si il n'en existe pas de valide -la propriété n'est donc pas vérifiée- , "Vrai" sinon (avec éventuellement la valeur du certificat trouvé).

Par exemple; l'usage pourra être : java testTSP <files> <mode> <longueurMaxi> avec comme modes (au moins) -verif, -nondet, -exhaust.

Vous avez à votre disposition des exemples de taille variable. Vous pouvez aussi en trouver par exemple à .

Attention! Ne pas utiliser la version "exploration exhaustive" sur des problèmes de grande taille!

2 Réductions polynomiales

On va maintenant illustrer la notion de réduction polynomiales d'une propriété à l'autre. 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. $ham : \{0 \dots n-1\} \to \{0 \dots n-1\}$, une **permutation** des n villes, telle que $D[ham(n-1)][ham(0)] = true$ et $D[ham(i)][ham(i+1)] = true$, 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. $ham : \{0 \dots n-1\} \to \{0 \dots n-1\}$, 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 4. $HamiltonCycle \leq_P TSP$

Q 4.1. Montrer que $HamiltonCycle$ se réduit polynomialement dans TSP.

Q 4.2.[*à coder*] Implémenter la réduction polynomiale de $HamiltonCycle$ dans TSP. Vous pouvez tester avec les données fournies.

Q 4.3. Qu'en déduire pour TSP?

Q 4.4. Pensez-vous que TSP se réduise polynomialement dans $HamiltonCycle$? Pourquoi?

Q 5. $HamiltonPath \leq_P HamiltonCycle$

Q 5.1. Montrer que $HamiltonPath$ se réduit dans $HamiltonCycle$.

Q 5.2.[*à coder*] Implémenter une réduction polynomiale $HamiltonPath$ dans $HamiltonCycle$.

Q 5.3. Montrer que $HamiltonPath$ se réduit dans TSP.

Q 6. *Composition de réductions*

En utilisant les deux réductions précédentes, implémenter une réduction de $HamiltonPath$ dans TSP.

3 Optimisation versus Décision

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 de longueur l, i.e. $tour : \{0 \dots n-1\} \to \{0 \dots n-1\}$, une **permutation** des n villes telle que $D[tour(n-1), tour(0)] + \sum_{i=0}^{n-2} D[tour(i), tour(i+1)] = l$.

TSPOpt2

Donnée

n, un entier – un nombre de villes

D, une matrice (n, n) d'entiers – elle représente les distances, qu'on suppose enti7res

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

Q 7. Montrer que si $TSPOpt1$ (resp. $TSPOpt2$) était P, la propriété TSP le serait aussi ; qu'en déduire pour $TSPOpt1$ (resp. $TSPOpt2$)?
Q 8. Montrer que si la propriété TSP était P, $TSPOpt1$ le serait aussi.
Q 9. *Plus dur...* Montrer que si la propriété TSP était P, $TSPOpt2$ le serait aussi.

Exemple d'architecture de code

Voici un exemple d'architecture qui **n'est qu'un exemple. Elle correspond aux sources "à compléter" donnés mais peut être modifiée/améliorée.**

```
/*****************************************************************************************************
             EXEMPLE D'ARCHITECTURE DE CODE POUR LA PARTIE 1-  CE N'EST QU'UNE PROPOSITION
*****************************************************************************************************/
package classesPb;

//la classe abstraite des problèmes de décision...
public abstract class PbDecision {
public abstract boolean aUneSolution();
}

//la classe abstraite ExpTime
public abstract class ExpTime extends PbDecision{
public abstract boolean aUneSolution();
//doit etre de complexite temporelle au plus exponentielle
}

//La notion de Certificat
interface Certificat{
   //affiche un certificat
   public void display();

   //pour l'énumération on utilisera un ordre total sur les certificats
   //par ex. le constructeur initialisera au plus petit  élément

   //transforme le certificat en son successeur pour l'ordre
   //déclenche une erreur ou indéfini si le certificat ets le dernier
    public void suivant();

   //retourne True Ssi le certificat n'a pas de successeur pour l'ordre
   public boolean estDernier();

   //modifie aléatoirement la  valeur du certificat
   //chaque valeur doit pouvoir être générée par au moins une exécution
   public void alea();
```

```
  //réinitialise le certificat au plus petit pour l'ordre
  public void reset();

  //permet la saisie d'un certificat
  public void saisie();
}

//la classe des problèmes NP
public abstract class NP extends ExpTime{

//on doit pouvoir definir pour le pb la notion de  certificat
//Attention: on doit pouvoir borner polynomialement la taille du certificat
// par rapport a la taille du probleme
abstract public Certificat cert();

//Algo de verification d'un certificat pour le probleme
//c'est l'algo A de la definition de NP par les certificats
abstract public boolean estCorrect(Certificat cert);

//algo exponentiel de decision de la propriete basee sur l'enumeration des certificats
// NP est inclus dans EXPTime
 public boolean aUneSolution() {
// A completer
    }

//algo non deterministe  polynomial:
//si il existe une solution AU MOINS UNE execution retourne Vrai
//si il n'en existe pas TOUTES les executions retournent faux!
 public boolean aUneSolutionNonDeterministe() {
// A completer
    }
}

package travellingSalemansProblem;

...class TSP extends NP{
    ... int nb_villes; //nb de villes
    ... int Dist[][];  //la matrice de distances (donc à valeurs >=0, et avec les valeurs diagonales
   ...  int l: la longueur maxi autorisée d'une tournée

 public Pbl_TSP(int n, int D[][] ){
    //juste le constructeur A ECRIRE
    }
   //A compléter comme  vous le souhaitez!
    ...
}

//la notion de certificat pour le problème TSP
... class  CertificatTSP implements Certificat{
private PblTSP pb; //le problème associé au certificat
//A compléter
//implementer l'interface
}
```