Dhersin Jérôme
Malapel Florian

ACT TP 4 – Classe NP

Qu'est-ce qu'une propriété NP?

Travelling Salesman Problem
Données

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

1) Notion de certificat

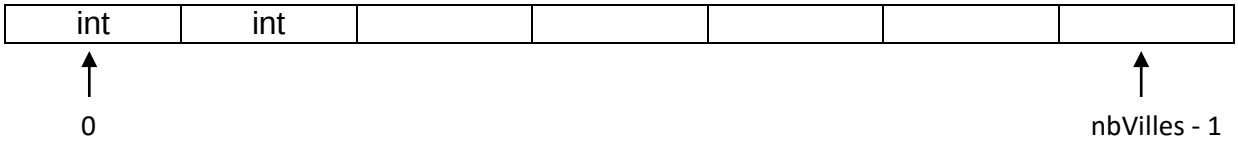
Une solution du problème est un ensemble de villes visitées par le voyageur.

Donc nous avons décidé que notre certificat sera un ensemble d'indices qui correspondront aux indices des villes.

Notre certificat est borné polynomialement par rapport aux données de l'entrée car la taille de l'ensemble (du certificat) dépend de la taille de l'entrée qui ici est le nombre de villes.

Pour l'implémentation ce sera donc un tableau d'entier. De taille maximale = nombre de villes.

La taille d'un certificat sera donc O(n) avec n = nombre de villes qui est une donnée du problème ce qui veut dire que la taille de notre certificat est borné par rapport à cette entrée.



Algorithme de vérification :

1. Vérifier si la taille du tableau est <= à la taille du tableau de l'entrée
2. Vérifier si le tableau ne contient pas de doublon
3. Vérifier si la somme des distances ≤ l

$$\sum_{0}^{nbVille-2} D[i][i+1] + D[nbVille-1][0] \leq longueur\_max$$

2.1) Algorithme de génération aléatoire d'un certificat

```
Pour i de 1 à nbVille

    Rand = random() ;   //   0 ≤ Rand ≤ nbVille - 1

    T[i] = Rand ;
```

Chaque ville a autant de chance qu'une autre d'apparaître dans la solution du certificat. Dans chaque case du tableau, chaque ville a autant de chance d'apparaître et donc tous les certificats ont la même probabilité d'apparaître.

2.2) Algorithme non déterministe polynomial

```
Certificat c = générer certificat aléatoirement ;

Si c est correct

        Afficher c ;

        Retourner vrai ;

Retourner faux ;
```

3.1) Notre certificat à une taille n (qui est égale au nombre de villes), chaque case peut prendre n valeurs, donc il y a n^n certificats possibles.

3.2) Enumération de tous les certificats

Pour passer d'un certificat à un autre on va changer une ville donc 1 indice dans une case.

Ordre des certificats : supposons nbVilles = 5

1er Certificat	**0000…..0**
2ème Certificat	0000…..1
3ème Certificat	0000…..2
……	…….
dernier Certificat	4444…..4

3.3) L'algorithme du British Museum

```
Certificat c = générer certificat;

Tant que c non Correct et c non dernier

        C = certificat suivant ;

Si c est correct

        Afficher c ;

        Retourner vrai ;

Retourner faux ;
```

On parcourt tous les certificats jusqu'à en trouver un correct.

Sachant que l'on peut avoir n ! certificats le parcourt de tous les certificats sera en O(n !)

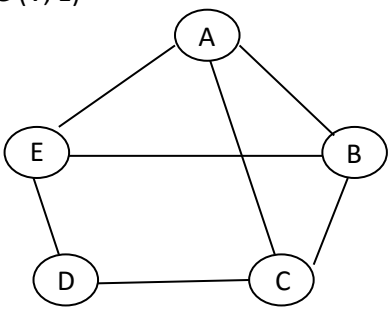
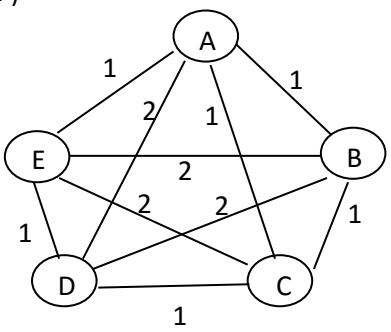
Réduction polynomiale

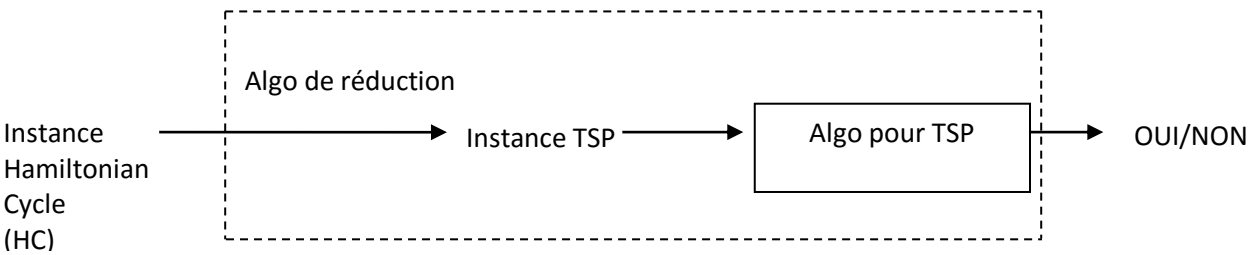
4.1) Hamilton cycle $\leq_p$ TSP

Pour chaque sommets u et v dans Hamilton Cycle, s'il y a une arrête entre les sommets on met 1 dans D[u][v] dans TSP sinon on met 0.

Problème Hamilton cycle	Problème TSP
N : entier (nbSommets) ----→	N : entier (nb Villes)
D [ ][ ] : bool(matrice des arrêtes) -----→	D [ ][ ] : entier(matrice des distances entre les villes)
----------→ = nbSommets	L : entier (longueur max)

Transformation des graphes :

Problème Hamilton Cycle	Problème TSP
G (V, E) 	G' (V, E') 
On ajoute (n*(n-1) - |A|) arrêtes avec n = nombre de sommets	



Algo pour Hamiltonian Cycle

Si HC se réduit dans TSP, alors TSP HC est plus facile que TSP.

Si pour chaque entrée de TSP et chaque entrée correspondante de HC l'algo pour TSP répond OUI, alors on peut dire que le problème HC se réduit polynomialement dans TSP.

Preuve :

Hamiltonian Cycle : G (V, E) TSP : G' (V, E')

Supposons un cycle Hamiltonien h dans G

- Chaque arrête e dans h et appartenant à E (arrêtes du problème Hamilton Cycle) -> coût = 1 dans G'
- H est un circuit Hamiltonien dans G' avec un coût de n (nombre de sommets) ce qui est une solution à TSP car par construction la longueur max de la tournée dans TSP vaut exactement le nombre de sommets.

Supposons que G' a un cycle Hamiltonien h' avec un coût de n(une solution de TSP)

- Si les arrêtes dans G' ont un coût de 1 ou 2, toutes les arrêtes de h' ont un coût de 1
- H' contient uniquement des arrêtes appartenant à E, car une arrête dans G' a un coût de 1 si cette même arrête est présente dans G
- Donc h' est un cycle Hamiltonien dans G qui passe par toutes les villes et qui a un coût de n

L'algorithme de transformation est borné par la taille de l'entrée du problème Hamilton Cycle, donc l'algorithme est polynomial en O ($|V|^2$) car on parcourt 1 seule fois la matrice des arrêtes pour créer la nouvelle matrice.

4.3)

- On sait que le problème HC (Hamiltonian Cycle) est NP-Complet et TSP est NP

- On sait que $HC \leq_p TSP$ donc ils ont la même classe de complexité TSP est au moins aussi dur que HC → TSP est NP-Dur

➔ TSP est NP-Complet car il est NP et NP-Dur

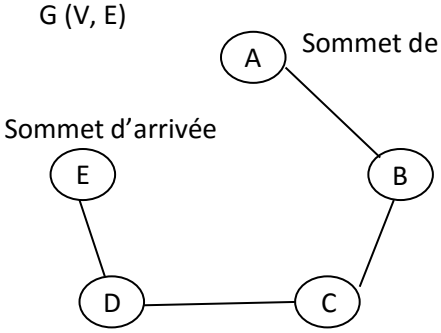
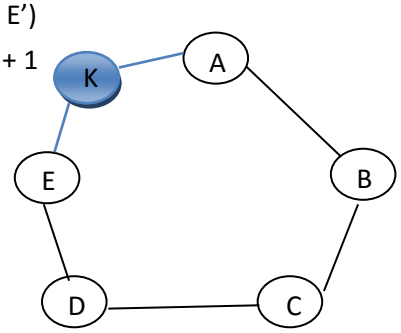
4.4) Oui on peut faire une réduction dans l'autre sens

TSP se réduit polynomialement en HamiltonCycle puisque que TSP est NP et HamiltonCycle NP-dur.

5.1) Hamilton path(HP) $\leq_p$ Hamilton cycle(HC)

Problème Hamilton path	**Problème Hamilton cycle**
N : entier (nbSommets) ---------→ +1	N : entier (nb Sommets)
D [ ][ ] : boolean(matrice des arrêtes) →+2 arrêtes	D [ ][ ] : boolean(matrice des arrêtes)

Transformation des graphes :

Problème Hamilton Path	Problème Hamilton Cycle
G (V, E) Sommet de départ Sommet d'arrivée  	G' (V', E') V' = V + 1 

Preuve :

Nous avons un graphe G qui contient un chemin Hamiltonien.

Un chemin Hamiltonien passe donc une et une seule fois par tout point du graphe ; si le graphe possède n points, un chemin Hamiltonien contient exactement n-1 arcs.

Supposons que l'on ait un chemin Hamiltonien h dans G qui part du sommet A pour arriver au sommet E, dans ce cas si on relie ces 2 sommets par un sommet intermédiaire, on obtient bien un chemin qui boucle sur lui-même → cycle h' dans G'.

Maintenant supposons que l'on a un cycle Hamiltonien h' dans G', si on supprime le dernier nœud, on obtient un chemin Hamiltonien h dans G.

L'algorithme de transformation est borné par la taille de l'entrée du problème Hamilton Path, donc l'algorithme est polynomial en O ($|V|^2$) car on parcourt 1 seule fois la matrice des arrêtes.

5.3) On sait que : $HC \leq_p TSP$ & $HP \leq_p HC$

Donc selon le théorème on peut en conclure que $HP \leq_p TSP$ car la réduction est transitive.

⇨ $HP \leq_p HC \leq_p TSP$

Optimisation versus décision

7)

a) TSPOpt1

Si TSPOpt1 était P, alors il existerait un algorithme déterministe polynomiale de résolution de TSPOpt1, resTSPOpt1 ().

Or s'il existe un n et une matrice D tel que l = TSPOpt1 (n, D) alors TSP (n,D,length) est vrai si et seulement si $l \leq$ length.

Ceci implique donc l'existence d'un algorithme polynomiale permettant de détermine si TSP est vrai ou non.

```
resTSP(n, D, length) {
        return resTSPOpt1 (n,D) ≤ length;
}
```

De ce fait TSP serait P aussi.

b) TSPOpt2

De même si TSPOpt2 était P, alors il existerait un algorithme déterministe polynomiale de résolution de TSPOpt2, resTSPOpt2 ().

On aurait alors un algorithme de résolution de TSP comme suit :

```
resTSP(n, D, length) {
        cert = resTSPOpt2(n,D);
        retourn cert.length() <= length;
}
```

Or cet algorithme est polynomial car resTSPOpt2 l'est.

De ce fait, TSP serait P.

8)

Supposons TSP P, et resTSP, un algorithme de rŽsolution de TSP dŽterministe polynomiale.

```
resTSPOPT1 (n, D) {
        int length = n*max(D); // max(D) retourne la plus grande valeur de D
        while(TSP(n,D,length) {
                length--;
        }
        return length+1;
}
```

Cet algorithme retourne en un temps polynomiale de n un résultat pour TSPOpt1.

9)

```
resTSPOPT2 (n, D) {
        int currentLine = 0;
        for (int i = 0; i< n; i++) {
                result.add(currentLine);
                for(int j = 0 ; j < n ; j++) {
                        if (D[currentLine][j] != infinite) {
                                currentLine = j;
                                break;
                        }
                }
        }
        return result;
}
```

Si TSP est P alors TSPOpt1 l'est aussi ce qui implique que \verb?resTSPOpt1 est déterministe polynomiale en n.

Et si resTSPOpt1 est polynomiale en n alors resTSPOpt2 l'est aussi.

De ce fait TSPOpt2 est P.