

Algorithmes et structures de données avancés :
Examen 1ère session
 Problèmes NP complets

Durée : 1h30. Une feuille recto écrite à la main autorisée.

Question 1 *Définitions*

1. Que signifie le sigle **NP** (pour les classes de problèmes) ?
2. Quelle est la différence entre les problèmes **NP complet** et les problèmes **NP difficile** ?
3. Comment est-ce qu'on peut prouver si un problème est **NP complet**.

Question 2 *Graphes réguliers*

1. Qu'est-ce qu'un graphe **régulier** ?
2. Dessiner **un** graphe 1-régulier avec 6 sommets qui est non-connexe et qui contient une clique de taille 2.
3. Considérer la déclaration du type suivant :

```
type t_matrice_adjacence=array[1..10,1..10] of integer;
```

Ecrire une fonction `function regulier(A : t_matrice_adjacence) : integer` qui prend comme entrée une matrice de taille 10×10 . Cette fonction doit renvoyer le degré de chaque sommet **si** le graphe est régulier, est **-1 sinon**. *Remarque* : Renvoyer ne signifie pas "afficher à l'écran".

4. Quelle est la complexité de votre algorithme pour le cas général d'une matrice $N \times N$?

Question 3 *Conditions nécessaires et conditions suffisantes*

1. Donner une condition nécessaire mais pas suffisante pour qu'un graphe (sans boucle) soit complet.
2. Donner une condition suffisante pour qu'un graphe (sans boucle) soit complet.

Question 4 *Problèmes TSP (Traveling Salesman problem - problème du voyageur de commerce)*

Etant donné un graphe $G = (V, E, c)$ avec V un ensemble de $n = |V|$ sommets, E un ensemble d'arêtes, et $c(i, j) \in \mathbb{N}$ les distances entre i et j . Un circuit est donné par une permutation π sur $\{1, \dots, n\}$, et la distance totale D de ce circuit est

$$D = c(\pi(1), \pi(2)) + c(\pi(2), \pi(3)) + \dots + c(\pi(n-1), \pi(n)) + c(\pi(n), \pi(1))$$

Considérer maintenant les trois variantes du problème NP-complet **TSP** :

Variante 1 Pour une valeur B donnée, est-ce qu'il existe un circuit avec une distance totale D inférieure à B .

Variante 2 Calculer la distance totale du circuit le moins cher.

Variante 3 Déterminer le circuit le moins cher.

Considérer également le graphe $G_1 = (V, E, c)$ suivant :

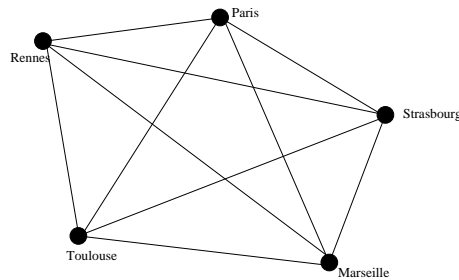


Figure 1: Le graphe $G_1 = (V, E)$.

avec tableau de distances entre les villes suivant :

c	Marseille	Paris	Rennes	Strasbourg	Toulouse
Marseille	0	775	1021	803	407
Paris	775	0	349	490	699
Rennes	1021	349	0	831	683
Strasbourg	803	490	831	0	1022
Toulouse	407	699	683	1022	0

1. Résoudre la variante 1 du problème **TSP** pour le graphe G_1 avec $B = 2630$.
2. Résoudre la variante 2 du problème **TSP** pour le graphe G_1 .
3. Résoudre la variante 3 du problème **TSP** pour le graphe G_1 .
4. Parmi ces 3 variantes, quels sont les problèmes d'optimisation ?
5. Prouver le Théorème 1 ci-dessous.

Théorème 1 *Si la variante 1 du problème **TSP** peut être résolue en temps polynomial par un algorithme non-déterministe, la variante 2 peut être résolue également en temps polynomial par un algorithme non-déterministe.*

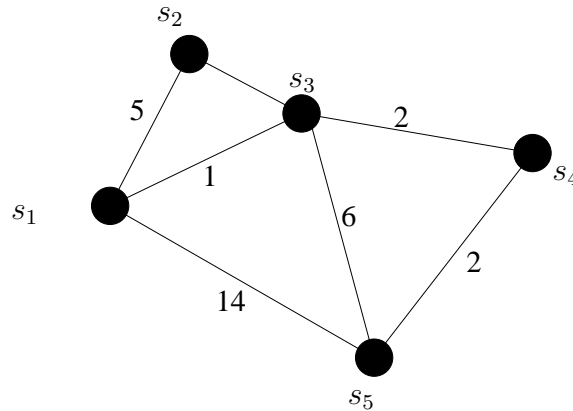
6. Prouver le Théorème 2 ci-dessous.

Théorème 2 *Si la variante 2 du problème **TSP** peut être résolue en temps polynomial par un algorithme non-déterministe, la variante 3 peut être résolue également en temps polynomial par un algorithme non-déterministe.*

7. Quel type de preuve avez-vous utilisé ? (*Remarque* : En cours, nous avons vu 3 types de preuves différentes).

Question 5 *Algorithme de Dijkstra*

Considérer le graphe G_2 suivant :



1. Faites tourner l'Algorithme de Dijkstra sur le Graphe G_2 avec $s = s_1$ dans le tableau suivant (Copier le tableau sur vos feuilles d'examen).

Ensembles			Distances				
M1	M2	M3	s_1	s_2	s_3	s_4	s_5
...							

2. Dessiner l'arbre recouvrant à partir de vos résultats.

Question 6 *Algorithme de Kruskal*

1. Dessiner l'**arbre minimal** couvrant du Graphe G_2 de l'exercice précédent avec l'Algorithme de Kruskal. Utiliser pour ceci le tableau suivant (Copier le tableau sur vos feuilles d'examen).

i	e[i]	$\phi(e[i])$	dedans	F
1	(s_1s_3)	1	✓	$\{(s_1s_3)\}$
...				

2. Quelle est la somme des poids des arêtes de l'arbre minimal couvrant T ?

Question 7 *Cryptographie*

1. Quel est le principal avantage des algorithmes de cryptage à clé asymétrique ?
2. Quel algorithme de cryptage à clé asymétrique connaissez-vous ?
3. Expliquer le principe de fonctionnement de cet algorithme.

Question 8 *Fonctions définies par récurrence*

1. Ecrire une fonction définie par récurrence `function factoriel(n : integer) : integer` qui calcule le factoriel du nombre entier n .