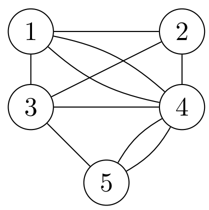
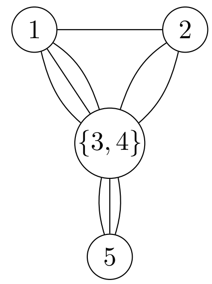
I Coupe dans un graphe

Soit $G = (S, A)$ un **multigraphe**, c'est-à-dire un graphe où il peut y avoir plusieurs arêtes entre les mêmes sommets.
Une **coupe** de G est une partition de S en S_1, S_2 avec $S_1 \neq \emptyset$ et $S_2 \neq \emptyset$. Sa **taille** est le nombre d'arêtes entre S_1 et S_2. Une coupe (S_1, S_2) est dite **minimum** si elle est de taille minimum parmi toutes les coupes de G.
Soient $s \neq t$ deux sommets de S. La **contraction** de $\{s, t\} \in A$ dans G, notée $G/\{s, t\}$, est le multigraphe formé à partir de G où s et t ont été fusionnés, où les arêtes entre s et t ont disparu, et toute arête entre s et un sommet u devient une arête entre $\{s, t\}$ et u (et de même pour une arête entre t et u).

Exemple : Un multigraphe et sa contraction de $\{3, 4\}$:





On cherche à trouver une coupe minimum dans G par un algorithme randomisé.

1. Donner une coupe minimum sur le graphe de l'exemple (à gauche).
2. On suppose qu'il existe k coupes minimums. On tire aléatoirement et uniformément une coupe. Déterminer la probabilité qu'elle soit minimum en fonction de $n = |S|$ et k. Commenter.

On propose l'algorithme suivant :

Début algorithme
- **Tant que** $|S| > 2$ **:**
  - Choisir $\{s, t\} \in A$ aléatoirement et uniformément.
  - $G \leftarrow G/\{s, t\}$.
- **Renvoyer** la coupe (S_1, S_2), où S_1 et S_2 sont les deux sommets restants de G

3. Appliquer cet algorithme sur le graphe en exemple, en choisissant arbitrairement les arêtes à contracter.
4. En détaillant les structures de données choisies, déterminer la complexité temporelle de cet algorithme.
5. Soit C une coupe minimum G. Soit k la taille de C. Montrer que $|A| \geq \dfrac{nk}{2}$.
6. Montrer que l'algorithme donne C avec probabilité au moins $\displaystyle\prod_{i=1}^{n-2}(1 - \frac{2}{n-i+1})$, puis exprimer plus simplement cette probabilité.
7. Combien de fois peut-on répéter l'algorithme précédent pour obtenir une probabilité supérieure à $\dfrac{1}{e}$ d'obtenir une coupe minimum ?
8. Déterminer le nombre maximum de coupes minimums dans un multi-graphe d'ordre n.

II Jeu à un joueur

On considère un **jeu** à un joueur, donné par un ensemble E (états possibles), d'un élément $e_0 \in E$ (état initial), d'une fonction $s : E \longrightarrow \mathcal{P}(E)$ ($s(e)$ est l'ensemble des états atteignables en un coup depuis e) et un sous-ensemble F (états gagnants) de E. Un état e_p est à profondeur p s'il existe une séquence finie $e_0, e_1, ..., e_p$ avec $e_{i+1} \in s(e_i)$ pour tout $i \in \{0, ..., p-1\}$. Si de plus $e_p \in F$, une telle séquence est une appelée une **solution**. Une **solution optimale** est une solution de profondeur minimale p^*. On admet qu'un même état peut être à plusieurs profondeurs différentes.

On considère le jeu suivant dans les 5 questions suivantes :

$$E = \mathbb{N}^*$$
$$e_0 = 1$$
$$s(e) = \{e + 1, 2e\}$$

1. Donner, sans justification, une solution optimale pour ce jeu lorsque $F = \{42\}$.
2. Quel algorithme du cours permettrait de trouver une solution optimale ?
3. Montrer que l'algorithme précédent est en complexité qui est exponentielle en p^*, en temps et en espace, c'est-à-dire de supérieure à α^{p^*} où α est une constante.
4. Décrire, en montrant qu'il est correct, un algorithme qui donne une solution optimale pour ce jeu en complexité linéaire en p^*.
5. Écrire en OCaml une fonction implémentant cet algorithme.

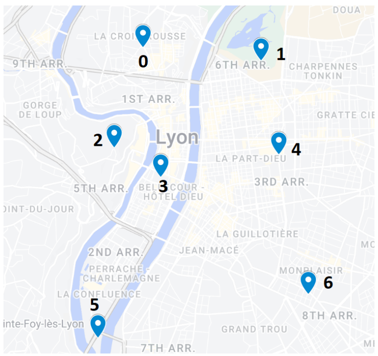
On considère maintenant un jeu quelconque.

6. En utilisant un parcours en profondeur modifié, expliquer comment obtenir une solution optimale en complexité $O(1)$ en mémoire. L'implémenter en OCaml.

III Algorithme de Christofides

On considère un graphe $G = (V, E)$ **non orienté**, **complet** et **pondéré** par une fonction $w : E \longrightarrow \mathbb{R}^+$. On supposera dans toute la suite que $V = \{0, ..., n-1\}$ où n est le nombre de sommets. Si G' est un graphe ou un ensemble d'arêtes, son **poids** $w(G')$ est la somme des poids des arêtes de G'.

On utilisera comme exemple le graphe G_{ex} suivant, représenté par matrice d'adjacence pondérée, dont les sommets sont des points d'intérêts de Lyon (numérotés de 0 à 6) et chaque arête correspond à une distance euclidienne entre les deux points :



$$G_{ex} = \begin{pmatrix} 0.0 & 2.3 & 1.75 & 2.2 & 2.9 & 4.8 & 5.1 \\ 2.3 & 0.0 & 2.9 & 2.6 & 1.5 & 5.4 & 4.0 \\ 1.75 & 2.9 & 0.0 & 0.9 & 2.9 & 3.2 & 4.2 \\ 2.2 & 2.6 & 0.9 & 0.0 & 2.1 & 2.9 & 3.2 \\ 2.9 & 1.5 & 2.9 & 2.1 & 0.0 & 4.4 & 2.5 \\ 4.8 & 5.4 & 3.2 & 2.9 & 4.4 & 0.0 & 3.7 \\ 5.1 & 4.0 & 4.2 & 3.2 & 2.5 & 3.7 & 0.0 \end{pmatrix}$$

On suppose que G est **métrique**, c'est-à-dire que w vérifie l'inégalité triangulaire : $\forall x, y, z \in V, \quad w(x, y) \leq w(x, z) + w(z, y)$

Un **cycle hamiltonien** dans un graphe est un cycle qui passe exactement une fois par chaque sommet. Dans la suite, on représente un cycle hamiltonien par sa liste de sommets, dans l'ordre (contenant donc chaque sommet exactement une fois).

Le **problème du voyageur de commerce (TSP)** consiste à trouver un cycle hamiltonien de poids minimum.

L'**algorithme de Christofides** donne une $\frac{3}{2}$-approximation du TSP. Voici son fonctionnement :

- Trouver un arbre couvrant de poids minimum T de G.
- Considérer V_1 l'ensemble des sommets de degré impair de T.
- Trouver un couplage parfait de poids minimum M de $G[V_1]$.
- Considérer le multigraphe H obtenu par union des arêtes de T et de M.
- Trouver un circuit eulérien C dans H.
- Transformer C en cycle hamiltonien en « sautant » les sommets répétés.

III.1 Recherche d'un arbre couvrant de poids minimum

1. Quel algorithme peut-on utiliser pour trouver un arbre couvrant de poids minimum dans un graphe ? L'appliquer sur G_{ex} en donnant les arêtes d'un arbre couvrant de poids minimum T_{ex} obtenu par l'algorithme, dans l'ordre.

III.2 Sommets de degré impair

Soit V_1 l'ensemble des sommets de degré impair dans T (en ne considérant que les arêtes de T).

2. Donner V_1 dans le cas où $T = T_{ex}$.
3. Montrer qu'un graphe G' non orienté possède un nombre pair de sommets de degré impair.

La question précédente montre donc que V_1 est de cardinal pair.

III.3 Couplage parfait de poids minimum

On définit $G[V_1]$ (**graphe induit** par V_1) comme le graphe dont l'ensemble de sommets est V_1 et dont deux sommets sont adjacents si et seulement si ils sont adjacents dans G. Les poids des arêtes de $G[V_1]$ sont les mêmes que ceux de G.

Un **couplage parfait de poids minimum** est un couplage parfait dont le poids est minimum parmi tous les couplages parfaits possibles.

4. Montrer l'existance d'un couplage parfait de poids minimum de $G[V_1]$.

5. On suppose que les sommets de G sont des points de $\mathbb{R}^2$ (comme c'est le cas pour G_{ex}). On peut alors considérer chaque arête comme un segment dans $\mathbb{R}^2$. Montrer que si M est un couplage de G dont deux arêtes se croisent (c'est-à-dire que les segments correspondants s'intersectent), alors M n'est pas de poids minimum.

6. Avec $G = G_{ex}$ et $T = T_{ex}$, donner les arêtes d'un couplage parfait de poids minimum M_{ex} de $G[V_1]$, ainsi que son poids.

III.4 Cycle eulérien

Un **multigraphe** est comme un graphe, sauf qu'il peut y avoir plusieurs arêtes entre deux sommets.
On définit le multigraphe $H = (V, E_H)$ dont les sommets sont les mêmes que G et tel que E_H contienne l'union des arêtes de T et de M (avec répétition : E_H est un multiensemble).

7. Dessiner H dans le cas où $G = G_{ex}$, $T = T_{ex}$ et $M = M_{ex}$.

Dans un graphe G' non orienté, un **cycle eulérien** est un cycle passant par chaque arête exactement une fois (mais qui peut passer plusieurs fois par un sommet, contrairement à un cycle hamiltonien).

8. Donner un cycle eulérien dans H dans le cas où $G = G_{ex}$, $T = T_{ex}$ et $M = M_{ex}$.

9. Montrer que si G' possède un cycle eulérien C, alors G' est connexe et tous les sommets de G' sont de degré pair.

On veut écrire un algorithme en OCaml permettant de trouver un cycle eulérien dans un graphe.

10. Écrire une fonction `supprime e l` supprimant la première occurence de `e` dans la liste `l`. Si `e` n'apparaît pas dans `l`, on renverra `l` inchangée. Par exemple, `supprime 3 [6; 3; 2; 3; 5]` doit renvoyer `[6; 2; 3; 5]`.

Dans la suite, on suppose que G' est connexe et que tous les sommets de G' sont de degré pair.

11. On part d'un sommet u quelconque de G', puis, tant que possible, on se déplace suivant une arête adjacente à u qui n'a pas encore été utilisée. Montrer que ce processus termine et que la liste des sommets visités forme un cycle C.

12. Écrire une fonction `cycle g u` qui renvoie le cycle C obtenu par l'algorithme précédent sous forme d'une liste de sommets, où G' est représenté par une liste d'adjacence `g`. On supprimera de `g` les arêtes utilisées par le cycle.

13. Montrer que G' possède un cycle eulérien, en raisonnant par récurrence.

14. En déduire une fonction `euler g u` qui renvoie un cycle eulérien (sous forme d'une liste de sommets) en partant depuis le sommet `u` dans le graphe G' donné sous forme de liste d'adjacence.

III.5 Cycle hamiltonien

Le cycle eulérien C dans le graphe H de la section précédente peut passer plusieurs fois par le même sommet. Pour le transformer en cycle hamiltonien, on supprime les sommets apparaissant plusieurs fois dans C (à part la première occurence).

15. Donner le cycle hamiltonien obtenu avec $G = G_{ex}$ ainsi que son poids. Comparer avec le cycle hamiltonien de poids minimum.

16. Écrire une fonction `supprime_doublons l` qui renvoie la liste obtenue en supprimant les doublons dans la liste de sommets `l`, sauf la première occurence de chaque sommet.
Par exemple, `supprime_doublons [1; 0; 1; 4; 6; 4]` doit renvoyer `[1; 0; 4; 6]`.

III.6 Preuve de la $\frac{3}{2}$-approximation

Soit C^* un cycle hamiltonien de poids minimum de G et T un arbre couvrant de poids minimum de G.

17. Montrer que $w(T) \leq w(C^*)$.

Soit V_1 l'ensemble des sommets de degré impair de T. Soit C_1^* un cycle hamiltonien de poids minimum de $G[V_1]$.

18. Montrer qu'on peut séparer les arêtes de C_1^* en deux couplages parfaits M_1 et M_2 de $G[V_1]$.

19. Soit M un couplage parfait de poids minimum de $G[V_1]$. Montrer que $w(M) \leq \min(w(M_1), w(M_2))$.

20. Montrer que $\min(w(M_1), w(M_2)) \leq \dfrac{w(C_1^*)}{2}$.

21. Montrer que $w(C_1^*) \leq w(C^*)$.

22. Soit C le cycle hamiltonien obtenu par l'algorithme de Christofides. Montrer que $w(C) \leq \dfrac{3}{2} w(C^*)$.