

# TP de Théorie des Graphes

## 1. Informations générales

- Section : Ingénierie Génie Logiciel (IGL3)

- Membres du groupe (Nom, Prénom) :

- Chaabane BOUSSADIA (IGL 3)
- Raed BETTAHER (IGL 3)
- Abderrahmen JEDIDI (IGL 3)

- Algorithme traité : Coloration des sommets (Algorithme glouton de Welsh et Powell)

- Langage utilisé : Python

## 2. Présentation de l'algorithme : Pourquoi on a besoin de tel algorithme ?

Soit une collection  $X = \{x_1, \dots, x_n\}$  de  $n$  tâches, chacune de durée indivisible d'un jour. Une série  $E$  de contraintes spécifie que certaines paires  $k$  de tâches ne peuvent pas être exécutées simultanément. L'objectif est de déterminer si toutes les tâches peuvent être exécutées dans un intervalle donné de  $k$  jours tout en respectant les contraintes, ou de trouver le plus petit intervalle de temps (nombre minimum de jours) nécessaire pour exécuter toutes les tâches. Ce problème peut être modélisé comme un problème de coloration de graphe, où chaque tâche est un sommet, chaque contrainte d'exclusion est une arête, et chaque couleur représente un jour où les tâches associées peuvent être exécutées simultanément.

L'algorithme de Welsh-Powell est une approche gloutonne pour résoudre ce problème de coloration de graphe. Il vise à assigner le plus petit nombre de couleurs (jours) possible tout en garantissant que deux sommets adjacents (tâches incompatibles) n'ont pas la même couleur.

- Fonctionnement général de l'algorithme de Welsh-Powell : soit un graphe  $G = (V, E)$ , où  $V$  est l'ensemble des tâches  $\{x_1, \dots, x_n\}$ , et  $E$  contient une arête entre  $x_i$  et  $x_j$  si ces tâches ne peuvent pas être exécutées simultanément .
  - 1) Tri des sommets : Triez les sommets de  $G$  par ordre décroissant de leur degré (nombre de voisins, càd nombre des contraintes associées à chaque tâche).
  - 2) Coloration gloutonne (d'une manière itérative jusqu'à ce que tous les sommets soient coloriés) :
    - a. Prenez le premier sommet non coloré dans l'ordre trié et assignez-lui la première couleur disponible (la plus petite couleur non utilisée par ses voisins déjà colorés).
    - b. Parcourez les sommets restants dans l'ordre, en assignant à chaque sommet non coloré la plus petite couleur possible qui n'est pas utilisée par ses voisins.

- Complexité des algorithmes :

Le tri des sommets par degré décroissant prend  $O(n \log n)$  [dans des algorithmes de tri avancé comme Tri Fusion ou le Tim Sort utilisé en Python)

Lors de la coloration des graphes, le meilleur cas est de colorier un graphe complet, sa complexité est égale à  $O(n)$ . Dans le pire de cas, cette étape prend  $O(n^2)$

La complexité totale est alors évaluée par l'expression  $O(n \log n + n^2) = O(n^2)$

NB : L'algorithme de Welsh-Powell est simple et rapide (par rapport aux autres algorithmes), mais il ne permet pas toujours de donner le nombre chromatique d'un graphe G.

Nombre chromatique : noté par  $\chi(G)$  est le plus petit nombre de couleurs nécessaires pour colorer les sommets de G de telle sorte que deux sommets adjacents (reliés par une arête) n'aient pas la même couleur.

### 3. Environnement et implémentation

- Langage de développement : Python (version 3.12) + Jupyter Notebook

- IDE : Visual Studio Code

- bibliothèques utilisées dans ce projet :

- networkx : Bibliothèque pour la création , la manipulation et de l'analyse de graphes et réseaux .
- matplotlib.pyplot : un module de bibliothèque Matplotlib pour créer des visualisations graphiques
- numpy : bibliothèque pour le calcul numérique , offrant des structures de données comme des tableaux multidimensionnels et des fonctions mathématiques performantes.
- Random : module standard de Python pour générer des nombres pseudo-aléatoires et effectuer des opérations aléatoires
- Pulp : bibliothèque pour la programmation linéaire, permettant de modéliser et résoudre des problèmes d'optimisation (Dans notre cas c'est la minimisation du nombre de couleurs dans un graphe)

- Structures de données principales :

Graphe : la bibliothèque NetworkX représente un graphe comme dictionnaire des dictionnaires pour stocker les sommets, les arêtes et les attributs.

Composants de base de cette structure :

- Sommets (nœuds) : représentés par des clés dans un dictionnaire. Chaque sommet peut être associé à des attributs stockés comme paires clé-valeurs.
- Arêtes : stockées dans un dictionnaire imbriqué où chaque sommet est mappé à un dictionnaire de ses voisins, avec des attributs optionnels

- Explication des étapes clés de code :

- Développement et test des méthodes de création et affichage des structures des données : On a utilisé la bibliothèque networkx pour créer des graphes à partir de nombre des sommets et une liste des tuples montrant les arêtes. Puis on a utilisé la bibliothèque matplotlib pour l'affichage des graphes
- Implémentation de l'algorithme de Welsh-Powell en utilisant le principe suivant : la fonction retourne la liste des couleurs de chaque nœud ainsi que le nombre des couleurs utilisés pour cette solution. On a aussi implémenté une fonction d'affichage de graphe colorié qui prend le graphe et les couleurs comme paramètres
- Test et évaluation des méthodes par des exemples
- Analyse des limites de l'algorithme de Welsh et Powell lors de détermination de nombre chromatique d'un graphe G avec un cas particulier et un exemple de code dont on a utilisé une bibliothèque de programmation linéaire pour l'implémenter.

#### 4. Jeux de tests

1<sup>er</sup> exemple : Une méthode d'organisation des examens pour IGL3 à la faculté des sciences de Tunis

Problématique : Lors de dernière session principale des examens des étudiants ingénieurs en génie logiciel 3. La plupart des étudiants ont trouvé le calendrier des examens non équilibré :

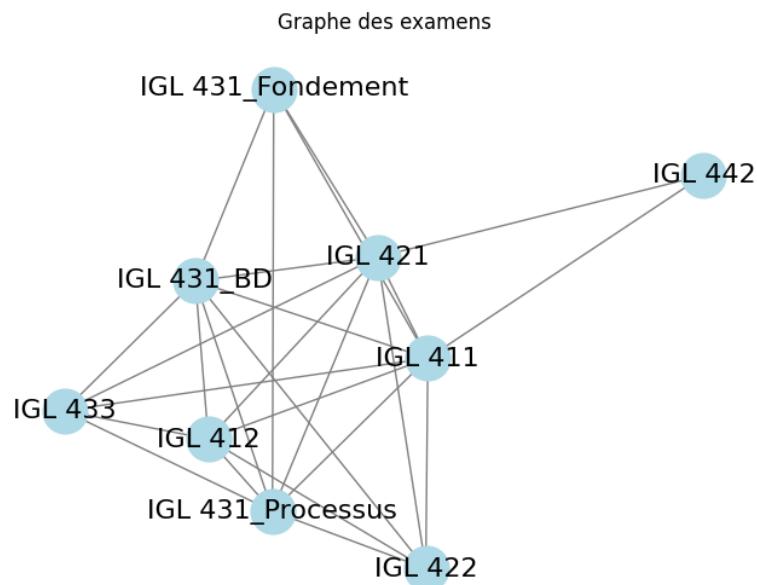
		FACULTE DES SCIENCES DE TUNIS Calendrier d'Examens						R-SAA-F-07   V1	IGL 3	
1		Ingénieur en Informatique: Génie Logiciel Première Année   Examens						22/05/2025		
Session Principale Mai 2025	Début des épreuves 08H30	Deuxième semestre								
		Jeudi	Vendredi	Samedi	Lundi	Mardi	Mercredi	Jeudi		
		15/05/2025	16/05/2025	17/05/2025	19/05/2025	20/05/2025	21/05/2025	22/05/2025		
		IGL 422	IGL 433	IGL 412						
	Graphes et flots	UX/UI Design	Systèmes et applications réparties	Fondement de l'IoT						
	Yous Smaïa	Ines Itabi	Hasshem Abbes	Chaker Essid						
	Durée	1h30	1h30	1h30	1h30	IGL 431	IGL 431	IGL 442	IGL 411	
				BD Avancées	Processus développement	Intro. à la macroéconomie	Algorithmique répartie	Algorithmique Numérique	IGL 421	
				Rihab Merimi	Amina Azzedine	Soumaya Jabbalah	Amira Ouerghi	Noumen Khouari		
	Durée			1h30	1h30	1h30	1h30	1h30	1h30	
	Groupe	Salle	Situé à	Capacité	Code Salle					
IGL3 GEX1:35 Etudiants	A. INFO	DPT INFO		40						
IGL3 GEX2: 22 Etudiants	P16	DPT PHYS		25						
Coordinateur	Anissa Ouerghi									

La notation "GREX" signifie groupe d'examen.  
Pour connaître votre groupe, consultez l'affichage.

C'est-à-dire que les matières sont réparties pour tenir compte la difficulté de la matière mesurée par son coefficient et le volume horaire étudié d'après le plan de cours.

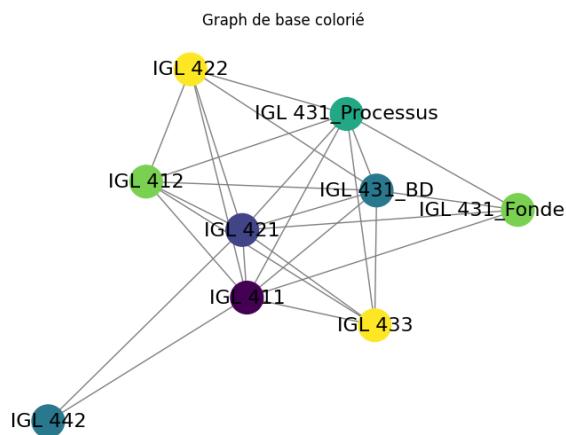
Par exemple, la matière de macroéconomie est de coefficient 1 et de difficulté inférieure par rapport les autres matières, mais les étudiants passent un jour juste pour passer cet examen. Par contre, au samedi, ils passent deux matières de difficulté grande au même jour : Systèmes et Applications réparties, BD avancée.

Suite à cette problématique, on a décidé d'utiliser l'implémentation de Welsh et Powell pour générer un calendrier des examens. On définit le graphe suivant :



Chaque nœud représente une matière, et un arête relie deux matières ayant une condition spécifique selon le score de difficulté de matière.

Résultat de l'exécution :



D'après ce graphe on peut répartir les examens de la manière suivante (exemple) :

- Jeudi 15/5 : IGL422 (Graphes et flots) , IGL433 (UX/UI Design)
- Vendredi 16/5 : IGL421 (Algorithme numérique)
- Samedi 17/5 : IGL431\_Processus (Processus de développement)
- Lundi 19/5 : IGL412 (SAR) IGL431\_Fondement (IoT)
- Mardi 20/5 : IGL 411 (AR)
- Mercredi 21/5 : IGL431\_BD (BD avancée) IGL442(macroéconomie)

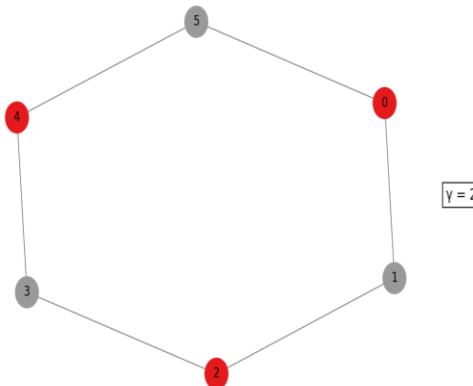
**Exemple 2 : SI G est un cycle alors  $\gamma(G) = 2$  si G est cycle pair sinon  $\gamma(G) = 3$**

- Cycle pair :

C'est un graphe cyclique avec 6 sommets (0 à 5) disposés en hexagone. Les couleurs utilisées sont le rouge et le gris.

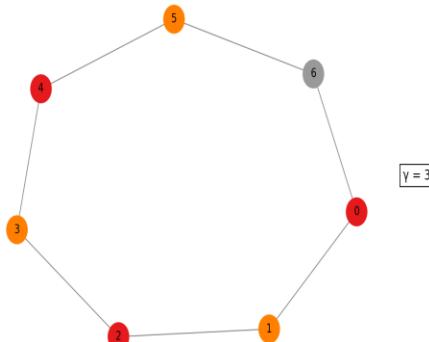
Puisqu'il s'agit d'un cycle pair (6 sommets), la propriété indique  $\gamma(G) = 2$  , ce qui correspond aux 2 couleurs utilisées. **La coloration est valide car les sommets adjacents ont des couleurs différentes**

Cycle Pair (6 sommets) - Couleurs utilisées: 2



- Cycle impair :

Cycle Impair (7 sommets) - Couleurs utilisées: 3

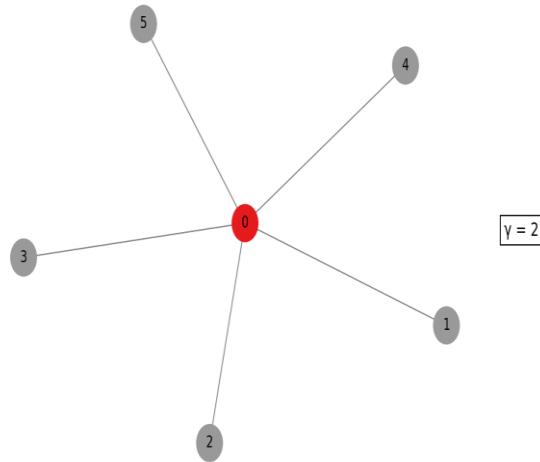


C'est un graphe cyclique avec 7 sommets (0 à 6) disposés en heptagone. Les couleurs utilisées sont le rouge, l'orange et le gris.

Puisqu'il s'agit d'un cycle impair (7 sommets), la propriété indique  $\gamma(G) = 3$ , ce qui correspond aux **3 couleurs utilisées**. La coloration est valide car les sommets adjacents ont des couleurs différentes.

- Graphe en étoile :

Graphe en étoile - Couleurs utilisées: 2

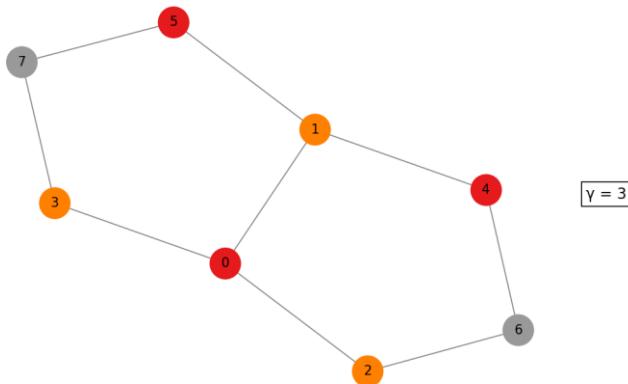


C'est un graphe en étoile avec 5 sommets (0 à 4), où un sommet central (0) est connecté à quatre sommets extérieurs. Les couleurs utilisées sont le rouge et le gris.

-> Le graphe n'est pas un cycle (c'est une structure arborescente). La propriété ne s'applique pas directement, mais **2 couleurs suffisent**, ce qui est cohérent pour un graphe biparti comme une étoile

- Graphe complexe :

Graphe complexe - Couleurs utilisées: 3

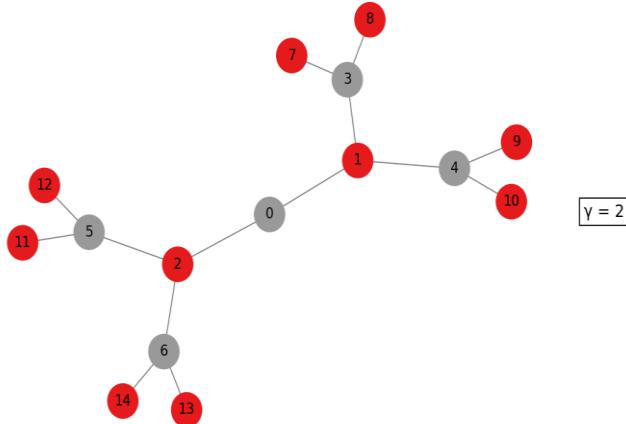


C'est un graphe complexe avec 7 sommets (0 à 6) et plusieurs arêtes les reliant, formant une structure non cyclique avec de nombreuses connexions. Les couleurs utilisées sont le rouge, l'orange et le gris.

**le graphe contient des sous graphes cycliques impairs => l'utilisation de 3 couleurs vérifie la propriété  $\gamma(G) = 3$**

- Arbre :

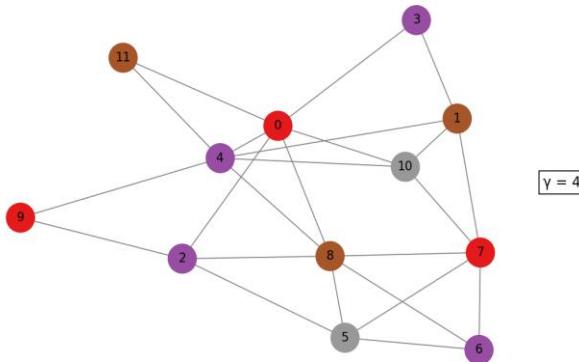
**Arbre (sans cycle) - Couleurs utilisées: 2**



C'est un graphe arborescent avec 14 sommets (0 à 13), présentant une structure hiérarchique sans cycles. Les couleurs utilisées sont le rouge et le gris. **2 couleurs suffisent, ce qui est attendu pour un arbre (nature bipartie).**

- Graphe aléatoire :

**Graphe aléatoire - Couleurs utilisées: 4**

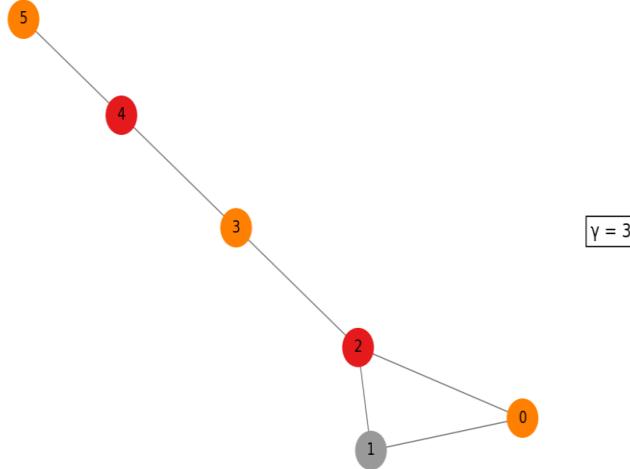


C'est un graphe aléatoire avec 11 sommets (0 à 10) et de nombreuses arêtes formant un réseau dense. Les couleurs utilisées sont le rouge, le violet, le marron et le gris.

L'utilisation de **4 couleurs** suggère un nombre chromatique plus élevé en raison des connexions denses, ce qui est cohérent avec la complexité d'un graphe aléatoire.

**Exemple 3 : Si  $K_n$  est un graphe complet alors  $\gamma(K_n) = n$**

**Graphe 1 - Manuel (Couleurs : 3)**

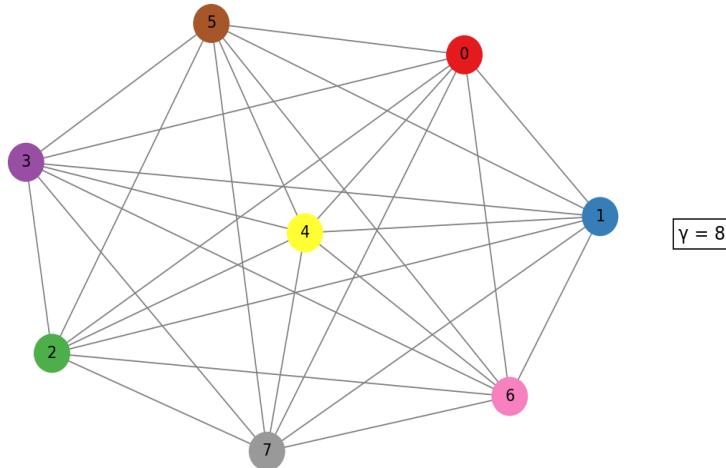


C'est un graphe avec 6 sommets (0 à 5) et des arêtes formant une structure linéaire principale (0-2-3-4-5) avec une branche (2-1 , 2 -0 , 0-1). Les couleurs utilisées sont le rouge, l'orange et le gris.

Il contient un sous graph complet  $(0,1,2)$   $K_3$  donc le nombre chromatique de ce graph est  $\geq 3$  et après la coloration on trouve  $\gamma(G) = 3$  ce qui est valide

- Graphe complet  $K_8$  :

**Graphe 2 - Complet  $K_8$  (Couleurs : 8)**

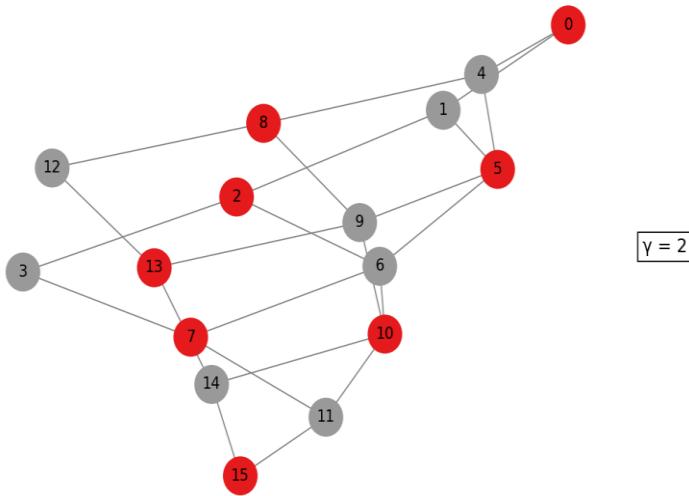


C'est un graphe complet K8 avec 8 sommets (0 à 7), où chaque sommet est connecté à tous les autres. Chaque sommet a une couleur différente : rouge, bleu, vert, violet, jaune, marron, rose et gris.

**Puisqu'il s'agit d'un K8, la propriété  $\gamma(K8) = 8$  ce qui correspond exactement aux 8 couleurs utilisées.** La coloration est valide car tous les sommets sont adjacents et nécessitent des couleurs différentes. La propriété est parfaitement respectée.

- Grille 4x4 :

**Graphe 3 - Grille 4x4 (Couleurs : 2)**

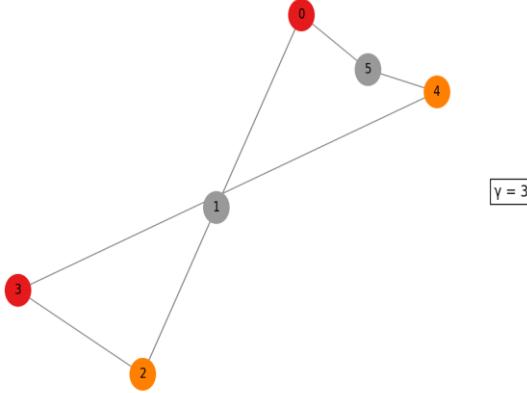


C'est une grille 4x4 avec 16 sommets (0 à 15), formant un réseau de 4 lignes et 4 colonnes. Les couleurs utilisées sont le rouge et le gris.

Une grille 4x4 est bipartie (on peut la colorier comme un échiquier). **Le plus grand sous-graphe complet est un K2 donc  $\gamma(K2)=2$ . Le graphe utilise exactement 2 couleurs**, ce qui correspond à la propriété. La coloration est valide car les sommets adjacents ont des couleurs différentes.

- Graphe Pondéré :

Graphe 4 - Pondéré (Couleurs : 3)

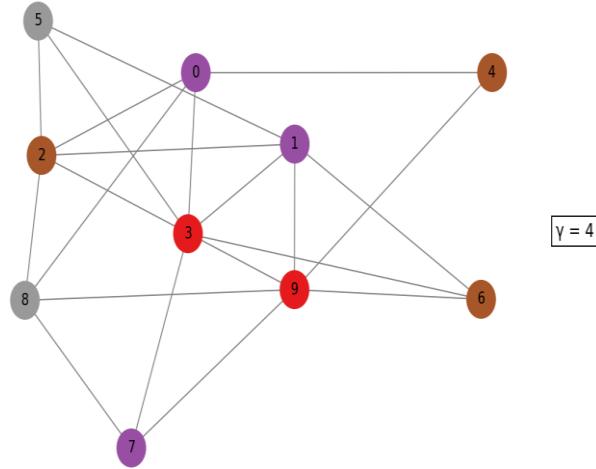


C'est un graphe avec 6 sommets (0 à 5) formant une structure non cyclique, avec des arêtes formant un chemin principal (0-1-2-3) et des connexions supplémentaires (0-4-5, 1-4). Les couleurs utilisées sont le rouge, l'orange et le gris.

**Le plus grand sous-graphe complet est un K3, donc  $\gamma(K3)=3$ . Le graphe utilise 3 couleurs, ce qui correspond à la propriété.** La coloration est valide car les sommets adjacents ont des couleurs différentes.

- Graphe Aléatoire :

Graphe 5 - Aléatoire (Couleurs : 4)



C'est un graphe aléatoire avec 10 sommets (0 à 9) et de nombreuses arêtes formant une structure dense. Les couleurs utilisées sont le rouge, le violet, le marron et le gris.

**Le plus grand sous-graphe complet est un K4 (par exemple, 0-1-3-9), donc  $\gamma(K4)=4$ . le nombre chromatique de G est donc => 4 .**

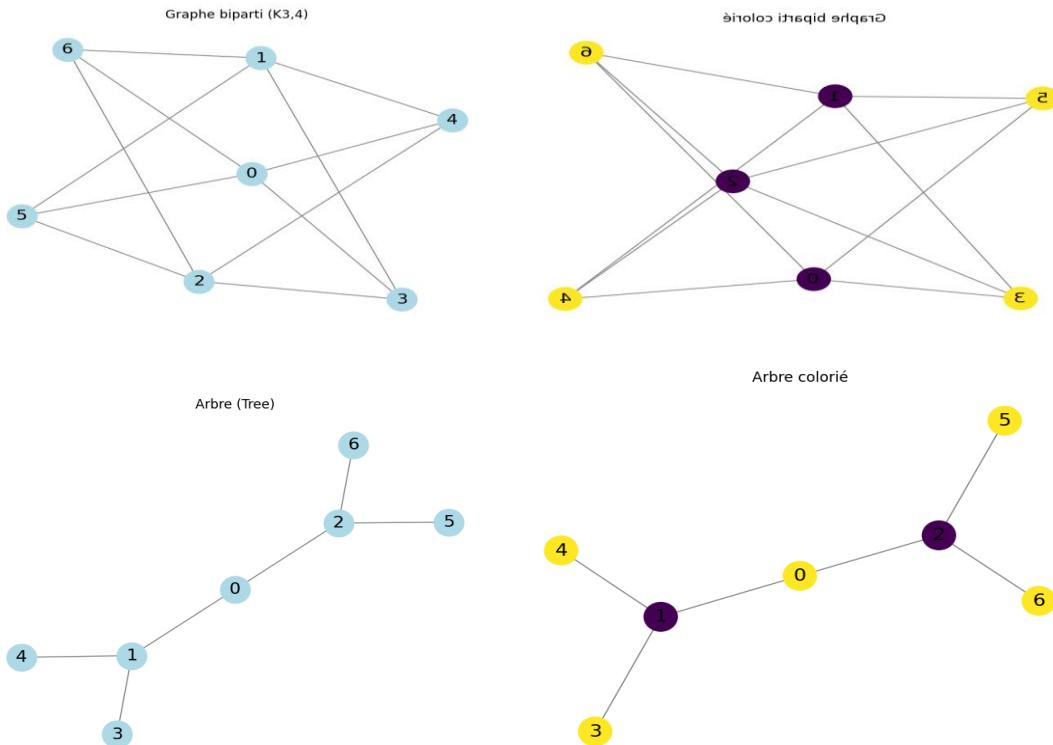
Après colorations G utilise 4 couleurs, ce qui est valide et minimal

#### Exemple 4 : Graphe biparti et arbre

##### ◆ Description du graphe

- **Nombre de sommets (biparti) :** 7
- **Nombre d'arêtes (biparti) :** 12
- **Nombre de sommets (arbre) :** 7
- **Nombre d'arêtes (arbre) :** 6

##### ◆ Capture d'écran



##### ◆ Résultat de l'exécution de l'algorithme Welsh-Powell

- **Biparti :**

Nombre chromatique : **2**

- **Arbre :**

- Nombre chromatique : **2**

◆ **Commentaire**

Le résultat confirme parfaitement la propriété théorique :

- Tout **graphe biparti** est **2-coloriable** ( $\gamma(G) = 2$ ).
- Tout **arbre** étant un graphe biparti, son **nombre chromatique** est aussi **2**.

**Exemple 5 : Graphe adjoint (Line Graph)**

◆ **Description du graphe**

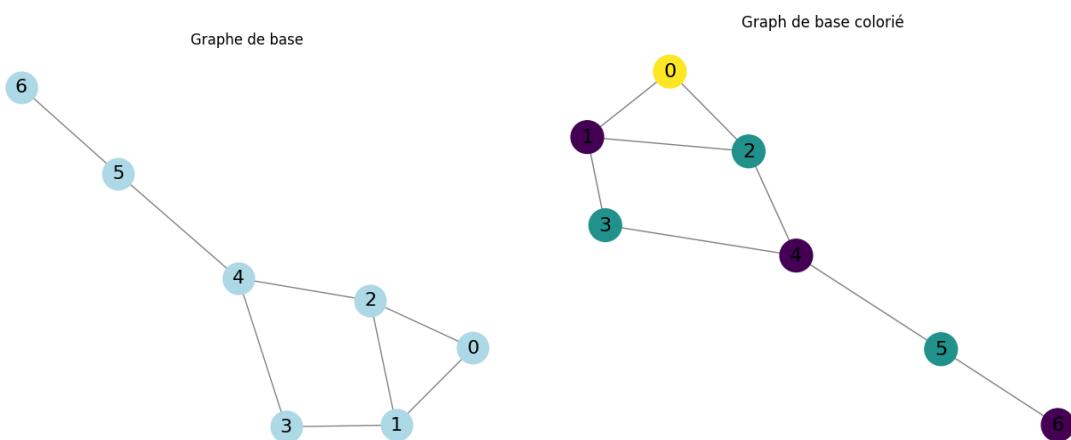
- **Graphe de base :**

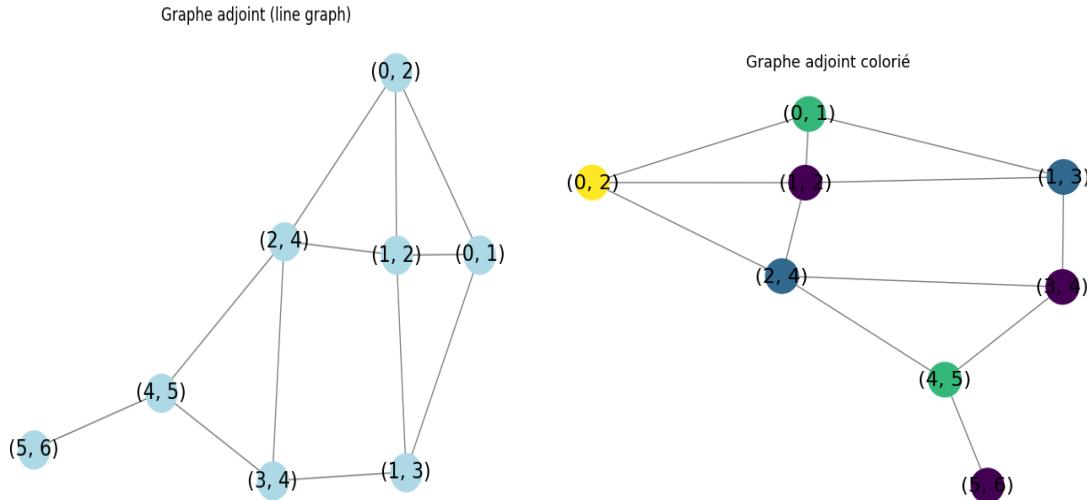
- **Nombre de sommets** : 7
- **Nombre d'arêtes** : 8

- **Graphe adjoint ( $L(G)$ ) :**

- **Sommets** : 8 (correspondant aux 8 arêtes du graphe de base)
- **Arêtes** : définies selon les arêtes adjacentes dans le graphe de base

◆ **Capture d'écran**





◆ **Résultat de l'exécution de Welsh-Powell**

- **Nombre chromatique de graphe adjoint: 4**

◆ **Commentaires**

Le maximum des degrés dans le graphe de base est : 3. Donc, l'indice chromatique qui correspond au nombre chromatique de graphe adjoint est comprise entre :

$$\Delta(G_{\text{base}}) \leq \chi(G_{\text{adjoint}}) \leq \Delta(G_{\text{base}}) + 1 \rightarrow 3 \leq \chi(G_{\text{adjoint}}) \leq 4$$

Et, dans notre cas, on a trouvé que le nombre chromatique de graphe adjoint (indice chromatique de graphe de base) peut être égale 4 (ou bien 3 si possible).

Ce qui est dans les normes de l'étude théorique

## 5. Discussion et limites

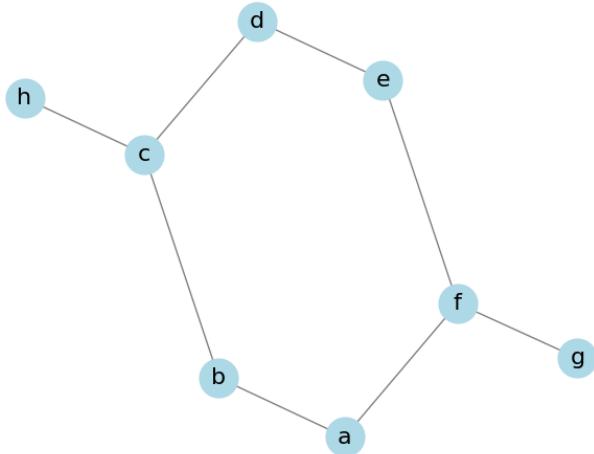
L'algorithme fonctionne dans tous les cas testés, tout en respectant les propriétés de chaque graphe mentionné (complet, biparti, cycle, ...) et le nombre des couleurs obtenu respecte l'encadrement théorique suivant

- Si on trouve un coloriage possible d'un graphe avec N couleurs, alors le nombre chromatique est nécessairement inférieur ou égal à N.
- Si le sommet de plus haut degré est r, alors le nombre chromatique est nécessairement inférieur ou égal à  $(r + 1)$

Cependant, cet algorithme ne permet pas de donner le nombre chromatique, c'est à dire le nombre des couleurs minimum pour colorier un tel graphe. Un tel contre-exemple se présente comme le suivant :

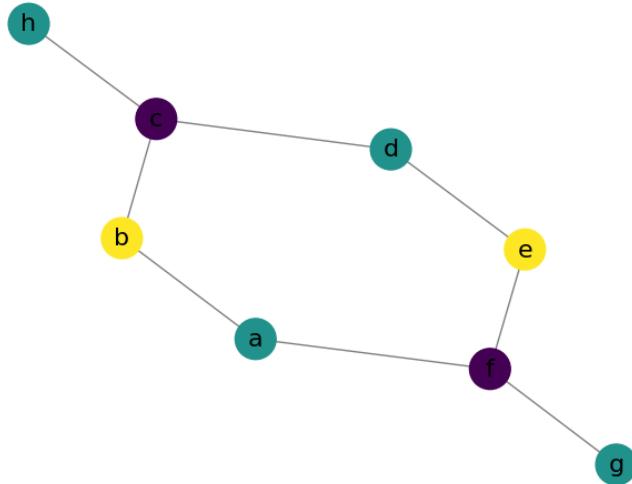
Considérons ce graphe suivant (résultat de code Python) :

Graphe de test pour ILP



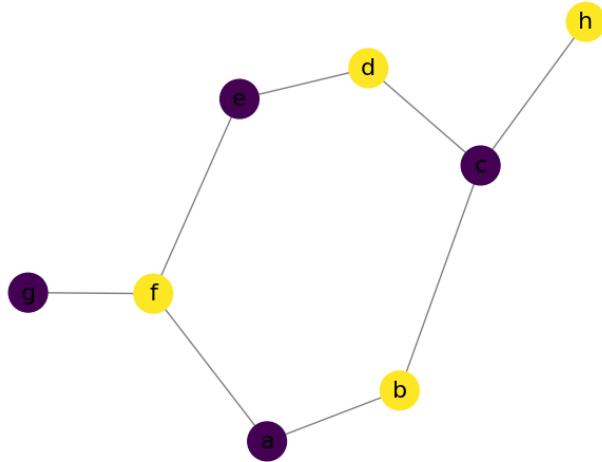
Si on utilise l'algorithme de Welsh et Powell , on se trouve dans le cas suivant :

Graphe colorié avec Welsh-Powell



L'algorithme a utilisé 3 couleurs pour colorier le graphe. Par contre si on utilise une approche linéaire optimisé pour minimiser le nombre des couleurs mais couteuse en temps d'exécution avec une complexité théorique exponentielle  $O(2^{n^2})$  on peut déduire le suivant :

Graphe colorié avec ILP



Il a utilisé que 2 couleurs seulement pour colorier le graphe ( $\chi(G) = 2$ )

- Amélioration et choix de l'algorithme adéquat :

Le choix d'algorithme nécessaire pour colorier un graphe dépend de plusieurs facteurs, notamment la taille du graphe et sa structure.

Pour les petits graphes ( $n \leq 50$ ) : il est recommandé l'ILP ou une approche qui utilise le backtracking pour une solution exacte

Pour les graphes de taille moyenne ( $50 \leq n \leq 1000$ ) : DSatur ou Welsh-Powell pour un bon compromis entre qualité et rapidité.

Pour les grands graphes : algorithmes gloutons ou métahéuristiques

Il faut aussi tenir compte de la structure des graphes qui peuvent faciliter la tâche. Par exemple, pour un graphe planaire on opte à utiliser un algorithme adapté aux 4 couleurs.

## 6. Conclusion

Dans ce TP, nous avons implémenté et analysé l'algorithme de Welsh-Powell pour la coloration des sommets d'un graphe, en utilisant Python. Cet algorithme glouton, qui trie les sommets par degré croissant avant d'assigner les couleurs de manière itérative, s'est relevé rapide et efficace lors des tests malgré qu'il ne donne pas le nombre chromatique exact dans certains cas.

Ce travail nous a permis de comprendre comment et pourquoi choisir un tel algorithme par rapport aux autres tous en étudiant la problématiques et les critères de graphe obtenu lors de la représentation de la structure de données.

## 7. Annexes

- Lien vers le dépôt du code : Vous trouverez le code ainsi que la trace d'exécution de ce TP dans ce répertoire GitHub suivante : [https://github.com/Chaabane2k03/Graphes\\_IGL3](https://github.com/Chaabane2k03/Graphes_IGL3)

- Instruction pour l'exécution de code :

- Assurer que vous avez installé Python dans votre machine ainsi que l'environnement de travail Jupiter.
- Si vous ne souhaitez pas l'installer ou vous avez rencontré des problèmes, vous pouvez exécuter le code à partir de la plateforme GoogleColab
- Exécuter le code, en cliquant sur Run All (l'installation des bibliothèques nécessaires est présente dans le code)

- Le rapport a été inspiré par l'article COLORATION DE GRAPHES : FONDEMENTS ET APPLICATIONS de RAIRO Operations Research