# Chapitre 15. Problèmes et algorithmes de coloration (hors-programme)

## 15.1 Coloration d'un graphe. Nombre chromatique

Un graphe est dit bien colorié si deux sommets voisins ont toujours des couleurs différentes. Un graphe ne peut pas être bien colorié s'il possède une boucle (il existerait alors un sommet voisin de lui-même), et le nombre d'arêtes qui relient deux sommets voisins est sans importance pour la coloration : on ne s'intéressera donc qu'aux graphes *simples*.

Le nombre minimal de couleurs nécessaires pour bien colorier un graphe $G$ est appelé nombre chromatique de $G$.

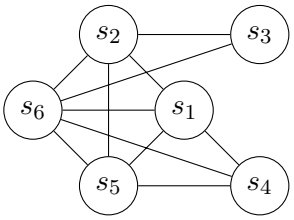
Dans les chapitres précédents on a vu des algorithmes efficaces pour tester si un graphe est connexe et pour construire des chemins dans des graphes. Pour les problèmes de coloration la situation est différente : en général *on ne connaît pas* d'algorithme *efficace*, sauf pour le problème de coloration avec deux couleurs — voir section 15.2. En particulier on ne connaît pas d'algorithme qui, appliqué à un graphe $G$ quelconque, calculerait rapidement son nombre chromatique ; on ne connait même pas de test efficace qui, appliqué à un graphe $G$ quelconque, déterminerait si trois couleurs suffisent pour colorier $G$.

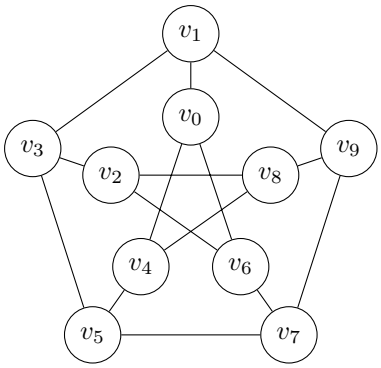
On conjecture depuis plusieurs dizaines d'années, sans savoir le démontrer, qu'en fait *il n'existe pas* d'algorithme efficace pour des problèmes tels que la coloration avec trois couleurs : c'est la conjecture P $\neq$ NP. L'énoncé précis de cette conjecture repose entre autres sur une définition précise (et indépendante de la technologie) de la classe des algorithmes efficaces, et sort du cadre de ce cours.

**Exercice 15.1.1** Écrire une fonction `bienColorie(G:graphe)` qui teste si un graphe est bien colorié.

**Exercice 15.1.2** Un graphe complet d'ordre $n$ est un graphe de $n$ sommets tel que chaque sommet est relié aux autres sommets par une arête ; on appelle $K_n$ un tel graphe. Dessiner $K_2$, $K_3$, $K_4$ et $K_5$. Quel est le nombre chromatique de $K_n$ ?

**Exercice 15.1.3** En remarquant que le graphe ci-contre contient des sous-graphes complets (lesquels ?) montrer que son nombre chromatique est supérieur ou égal à 4. Vérifier que l'on peut effectivement le colorier avec 4 couleurs.




**Exercice 15.1.4** Calculer le nombre chromatique du graphe de Petersen, représenté ci-contre à gauche.

**Exercice 15.1.5** Indiquer si les propositions suivantes sont vraies ou fausses en justifiant votre réponses :

1. Si un graphe contient un graphe complet $K_n$, son nombre chromatique est au moins $n$.
2. Pour montrer que le nombre chromatique d'un graphe est $n$, il suffit d'établir un coloriage avec $n$ couleurs.
3. Pour montrer que le nombre chromatique d'un graphe est $n$, il suffit de montrer qu'il contient un graphe complet $K_n$
4. Si le nombre chromatique d'un graphe est $n$, il contient forcément un graphe complet $K_n$.
5. Pour montrer que le nombre chromatique d'un graphe est au plus $n-1$, il suffit de montrer qu'il ne contient pas de graphe complet $K_n$.

**Exercice 15.1.6** Quel est le nombre chromatique d'une grille triangulaire $T_n$ (voir exercice 10.2.5 pour la définition) ?

Même question pour une grille carrée.

**Exercice 15.1.7** Un musée est constitué de 9 salles notées A, B, C, D, E, F, G, H et S. Le plan du musée est représenté sur la figure 15.1.

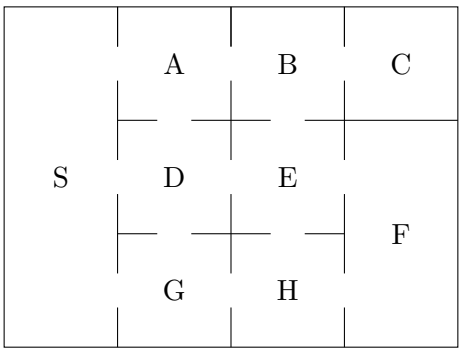



FIGURE 15.1 – Plan du musée

Pour rompre une éventuelle monotonie, le conservateur du musée souhaite différencier chaque salle de sa ou des salles voisines (c'est-à-dire accessibles par une porte) par la moquette posée au sol. Quel est le nombre minimum de types de moquettes nécessaires pour répondre à ce souhait ? Justifier.

**Exercice 15.1.8** Un aquariophile souhaite acquérir 6 espèces de poissons : A, B, C, D, E, F. Certaines espèces ne peuvent cohabiter dans un même aquarium. La liste des incompatibilités est la suivante : A avec E, F avec D, E avec F, C avec D, B avec C, A avec F, B avec D, B avec F. Il se pose la question de savoir de combien d'aquariums il doit disposer au minimum.

1. Dessiner le graphe représentant les incompatibilités entre les poissons.
2. Modéliser le problème posé en un problème sur le graphe précédent.
3. Résoudre le problème posé.

**Exercice 15.1.9** Un opérateur de télécommunications mobiles veut couvrir une zone géographique à l'aide de quelques bornes de téléphonie mobile. Pour éviter les interférences, on utilise différents canaux : deux bornes éloignées de moins de 600 m doivent utiliser des canaux différents. La figure 15.2 montre la position des différentes bornes sur une grille dont les carreaux sont de taille 100 m × 100 m.

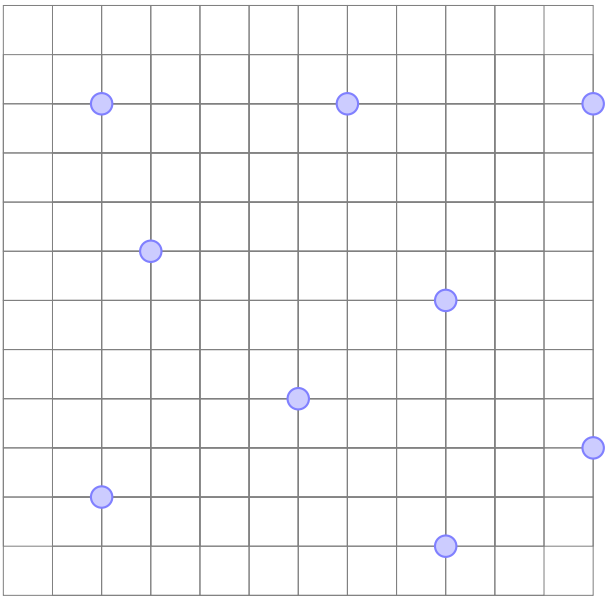


FIGURE 15.2 – Carte représentant la position de bornes de téléphonie

Quel est le nombre minimum de canaux que l'opérateur de télécommunications devra utiliser pour que son réseau de téléphonie mobile puisse fonctionner sans interférence ?

Si la borne en bas du bord droit avait été 100 m plus à gauche, est-ce que le résultat serait le même ? Justifier.

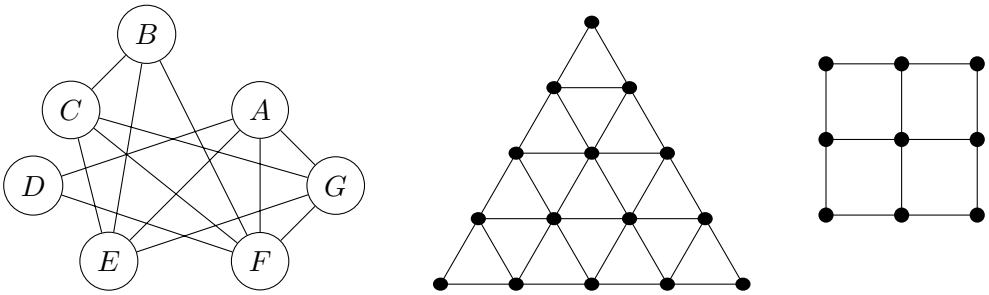
## 15.2 Graphes 2-coloriables

On dit qu'un graphe est 2-coloriable si son nombre chromatique vaut au plus 2, autrement dit s'il est possible de le colorier avec deux couleurs. Un tel graphe $G$ est aussi appelé *biparti*, ce qui signifie que l'on peut partitionner les sommets en deux ensembles $S_1$ et $S_2$ tels que les arêtes de $G$ relient toujours un sommet de $S_1$ à un sommet de $S_2$ — on peut donc attribuer la couleur $c_1$ à tous les sommets de $S_1$, et la couleur $c_2$ à tous les sommets de $S_2$.

Un cycle impair est un cycle de longueur impaire (voir chapitre 11), autrement dit un cycle avec un nombre impair d'arêtes (et donc un nombre impair de sommets si l'on ne compte pas deux fois le sommet de départ qui est aussi le sommet d'arrivée) ; le cycle impair le plus trivial est le triangle.

**Théorème 15.2** *Un graphe $G$ est 2-coloriable si et seulement si $G$ ne possède pas de cycle impair.*

**Exercice 15.2.1** Parmi les graphes suivant :




1. lesquels sont 2-coloriables ?
2. pour le ou les autres, calculer le nombre chromatique.

### Algorithme

Pour tester si un graphe connexe $G$ est coloriable avec deux couleurs données $c_1$ et $c_2$, on dispose d'un algorithme qui ressemble au test de connexité (voir section 11.5) :

1. choisir un sommet initial $s$ et le colorier avec la couleur $c_1$ ;
2. tant qu'il existe un sommet $t$ non coloré avec un voisin coloré :
   a) tester si les voisins de $t$ déjà coloriés ont tous la même couleur ;
   b) si c'est le cas, colorier $t$ avec l'autre couleur ;
   c) sinon, arrêter l'algorithme.

Cet algorithme prouve en même temps le théorème 15.2, c'est l'objet de l'exercice suivant.

**Exercice 15.2.2**

1. Montrer que lorsque cet algorithme colorie tous les sommets du graphe, celui-ci est bien colorié.
2. Montrer que lorsque l'exécution de l'algorithme se termine prématurément (étape 2.c), un cycle impair a été détecté — le graphe n'est donc pas 2-coloriable.

3. En déduire le théorème.

**Exercice 15.2.3**

1. Écrire une fonction `effacerCouleurs(G)` qui colorie en blanc tous les sommets de $G$.

2. Écrire une fonction `sommetColoriable(G)` qui renvoie un sommet de $G$ blanc ayant au moins un voisin non blanc et `None` si un tel sommet n'existe pas.

3. Écrire une fonction `monoCouleurVoisins(s)` qui, si les voisins non blancs du sommet $s$ sont tous de la même couleur, renvoie cette couleur et dans le cas contraire ($s$ a deux voisins de couleurs non blanches différentes) renvoie `None`.

4. Écrire une fonction `deuxColoration(G,c1,c2)` qui tente de colorier le graphe $G$ avec les deux couleurs $c1$ et $c2$ en utilisant l'algorithme détaillé plus haut, et renvoie `True` en cas de succès, `False` sinon.

5. Tester la fonction sur :

   a) des grilles — utiliser la fonction `construireGrille(m,n)`,

   b) des arbres — fonction `construireArbre(degre,hauteur)`,

   c) des graphes bipartis complets — fonction `construireBipartiComplet(m,n)` ; un graphe $G$ est biparti complet, s'il existe une partition de son ensemble de sommets en deux sous-ensembles $U$ et $V$ telle que chaque sommet de $U$ est uniquement relié à chaque sommet de $V$. Si $U$ est de cardinal $m$ et $V$ est de cardinal $n$ le graphe biparti complet est noté $K_{m,n}$ ;

   d) le graphe de Petersen,

   e) le cube, l'octaèdre et le dodécaèdre.

   Après chaque essai afficher le graphe pour vérifier soit qu'il est bien colorié, soit qu'un sommet non coloré $t$ possède deux voisins de couleurs distinctes ; dans ce dernier cas vérifier qu'un cycle impair relie $t$ et certains des sommets déjà coloriés.

*Conseils* : lorsque la fonction `deuxColoration` découvre un sommet $t$ non coloré avec deux voisins de couleurs différentes (cas d'un graphe non 2-coloriable), afficher ce sommet en ajoutant simplement une instruction `print(t)`.

Les graphes de cette section sont pour la plupart mieux dessinés en spécifiant l'algorithme de dessin comme suit : `afficherGraphe(G,algo='neato')`. Voir http://www.graphviz.org pour une description rapide des algorithmes disponibles avec ce logiciel de dessin de graphes : *dot, neato, fdp, twopi* et *circo*.

**Exercice 15.2.4** Écrire une fonction `areteEntre(s1:sommet, s2:sommet)` qui retourne une arête entre les sommets $s1$ et $s2$ s'il en existe une, `None` sinon. Comme dans l'exercice 11.5.10 modifier la fonction `sommetColoriable` pour marquer l'arête qui relie ce sommet non coloré à son voisin coloré ; ne pas oublier de modifier aussi la fonction `effacerCouleurs` pour démarquer les arêtes en même temps.

Tester à nouveau la fonction `deuxColoration` comme dans l'exercice précédent, et l'afficher à chaque fois le graphe : que remarque-t-on à propos du sous-graphe formé par les arêtes marquées (et leurs extrémités) ? Lorsque le graphe n'est pas 2-coloriable vérifier qu'il existe toujours un cycle impair dont toutes les arêtes, sauf une, sont marquées.

**Exercice 15.2.5** Utiliser la fonction `melange` comme dans l'exercice 11.6.1 pour modifier la fonction `sommetColoriable` afin que le résultat ne dépende plus de l'ordre dans lequel sont rangés les sommets du graphe, ni de l'ordre des arêtes incidentes à un sommet.

Une fois cette modification effectuée, appliquer plusieurs fois la fonction `deuxColoration` au dodécaèdre ; après chaque calcul afficher le graphe et observer quel est le sommet non coloré $t$ avec deux voisins de couleurs différentes qui a stoppé l'exécution de l'algorithme ; observer que le sommet $t$ est situé sur un cycle impair dont toutes les arêtes, sauf une, sont marquées, et que ce cycle n'est pas toujours un pentagone (bonne chance).

## 15.3 Graphes planaires

Un graphe est dit planaire s'il *existe* une façon de le dessiner dans le plan sans que ses arêtes se croisent. Voici un théorème très célèbre, car facile à énoncer et très difficile à démontrer :

**Théorème 15.3 (des quatre couleurs)** *Tout graphe planaire peut être «bien »colorié avec* 4 *couleurs.*

Voir wikipedia pour une excellente présentation de ce théorème et de son histoire étonnante (origine de la conjecture, preuves percées, emploi d'ordinateurs pour les preuves correctes).

On appelle carte planaire tout dessin d'un graphe dans le plan sans croisement d'arêtes ; à un graphe planaire correspondent en général beaucoup de cartes planaires distinctes. En plus des sommets et des arêtes une carte possède des faces, car elle découpe le plan en composantes connexes disjointes, bordées par les arêtes. On vérifie facilement qu'un dessin dans le plan correspond à un dessin sur la sphère, et vice-versa ; la seule différence est qu'il semble y avoir une face de plus sur la sphère, mais celle-ci correspond à la face externe dans le plan (la composante connexe qui « part vers l'infini »), qu'il ne faut pas oublier quand on compte les faces. Si $S, A, F$ désignent respectivement l'ensemble des sommets, l'ensemble des arêtes et celui des faces, et si la carte est connexe, on a la célèbre *formule d'Euler* :

$$|S| - |A| + |F| = 2$$

**Exercice 15.3.1** Soit $K_4$ le graphe complet à 4 sommets : dessiner $K_4$ de deux façons, l'une avec deux arêtes qui se croisent, l'autre sans croisement. $K_4$ est-il planaire ?

$K_5$ est-il planaire ? Quelles sont les valeurs de $n$ pour lesquelles $K_n$ est planaire ?

**Exercice 15.3.2** Montrer que le graphe de l'exercice 15.1.3 est planaire en proposant plusieurs cartes planaires distinctes pour ce graphe. Vérifier la formule d'Euler sur ces exemples.

**Exercice 15.3.3** On note $K_{n_1,n_2}$ le graphe biparti complet composé de deux ensembles de sommets $A$ et $B$ avec respectivement $n_1$ et $n_2$ éléments, et tel que tout sommet de $A$ soit relié à tout sommet de $B$. Étudier quelles sont les valeurs du couple $(n_1, n_2)$ pour lesquelles ce graphe est planaire.

**Exercice 15.3.4** On appelle degré d'une face le nombre d'arêtes qui la bordent : énoncer une formule analogue à celle des poignées de main (voir section 10.2) ; vérifier cette formule sur les cartes de l'exercice 15.3.2.

**Exercice 15.3.5** Démontrer la formule d'Euler par récurrence sur le nombre d'arêtes $|A|$ ; attention au cas particulier où la suppression d'une arête déconnecte la carte.

**Exercice 15.3.6** Soit $G$ un graphe planaire simple : $G$ ne possède ni boucle ni arête multiple, ce qui n'est pas restrictif quand il s'agit de le colorier — *cf.* introduction de ce chapitre. Montrer que $G$ possède au moins un sommet de degré au plus 5. *Conseils* : raisonner par l'absurde en supposant que tous les sommets sont de degré supérieur ou égal à 6, et appliquer la formule des poignées de main, et utiliser la formule vue à l'exercice 15.3.4 ; remarquer aussi que toute face (d'une carte planaire associée à $G$) est de degré au moins 3 (car le graphe est simple), ajouter une cuiller de formule d'Euler, et bien mélanger ...

**Exercice 15.3.7** Dessiner un graphe planaire simple sans sommet de degré inférieur à 5 (bonne chance).

## 15.4 Heuristique

Heuristique est un terme dérivé d'une expression grecque signifiant « *discours propre à découvrir* », par opposition à « *discours propre à convaincre* », selon le *Trésor de la Langue Française* : http://atilf.atilf.fr. En sciences on désigne ainsi une méthode ou un principe (souvent simples et astucieux) qui permettent d'obtenir une solution approchée d'un problème, ici celui de la coloration d'un graphe, pour lequel on ne connaît pas de méthode à la fois efficace et rigoureuse.

Voici un algorithme qui permet de colorer un graphe $G$ avec peu de couleurs, mais qui ne garantit pas que l'on ne pourrait pas faire mieux, c'est-à-dire utiliser moins de couleurs ; $G$ est supposé simple (cf. introduction de ce chapitre), ainsi le degré d'un sommet est égal à son nombre de voisins :

1. ordonner les couleurs disponibles dans un ordre arbitraire ;
2. choisir un sommet $s$ de degré minimal $d$ ;
3. retirer $s$ (et toutes les arêtes issues $s$) du graphe $G$, ce qui fournit un graphe $G'$ ;
4. colorier $G'$ par la même méthode ;
5. $s$ est alors entouré de $d$ voisins colorés : colorier $s$ avec la première couleur autorisée, c'est-à-dire avec la première couleur qui ne fait pas partie de l'ensemble des couleurs des voisins de $s$.

**Exercice 15.4.1** Exécuter l'algorithme à la main sur le graphe de l'exercice 15.1.3. Autres exemples simples : grille rectangulaire $3 \times 3$, grille triangulaire $T_3$ (voir exercices 10.2.4 et 10.2.5), octaèdre.

**Exercice 15.4.2** Soit $G$ un graphe connexe 2-coloriable ; montrer par récurrence sur le nombre de sommets que l'algorithme ci-dessus colorie $G$ avec deux couleurs, à condition que la suppression répétée de sommets de degrés minimaux ne déconnecte jamais le graphe.

Donner un exemple de graphe $G$ connexe 2-coloriable et possédant un sommet $s$ de degré minimal dont la suppression déconnecte $G$ (le graphe $G'$ n'est plus connexe) ; montrer que dans ce cas l'algorithme peut échouer à colorier $G$ avec deux couleurs. Proposer une modification de l'algorithme qui corrige ce défaut.

**Exercice 15.4.3** Comme on ne dispose pas de moyen de supprimer un sommet dans un graphe, on va programmer cette heuristique en marquant les sommets, avec la convention qu'un sommet *marqué* sera considéré comme *supprimé* :

1. écrire une fonction `nbVoisinsNonMarques(s:sommet)` qui retourne le nombre de voisins du sommet $s$ qui ne sont pas marqués ;
2. écrire une fonction `sommetNonMarqueMinimal(G:graphe)` qui retourne un sommet non marqué du graphe $G$ ayant un nombre minimal de voisins non marqués ;
3. écrire une fonction `couleurLibre(s:sommet)` qui retourne la première couleur différente des couleurs actuelles des voisins de $s$ ;
4. utiliser les fonctions `sommetNonMarqueMinimal` et `couleurLibre` pour écrire une fonction `degmin(G:graphe)` qui colorie un graphe $G$ selon l'algorithme décrit dans cette section ;

5. terminer en écrivant une fonction `colorierGraphe(G:graphe)` qui efface les couleurs et les marques du graphe $G$ avant de lancer `degmin` ;

6. améliorer la fonction précédente pour qu'elle retourne le nombre de couleurs employées.

*Conseils* :

a) Il est possible qu'il n'existe pas de sommet non marqué ! Ceci signifie que tous les sommets ont été supprimés, et dans ce cas la fonction `sommetNonMarqueMinimal` doit retourner `None`.

b) La fonction `degmin` supprime le sommet $s$ retourné par `sommetNonMarqueMinimal`, c'est-à-dire qu'elle le marque, après quoi elle appelle ... `degmin` pour colorier le graphe privé de $s$. Une fonction qui s'appelle elle-même est dite *récursive* ; pour toute fonction de ce type il faut spécifier un cas particulier sans appel récursif, sinon on entre dans une boucle sans fin. Ici le plus simple est de tester si $s$ est égal à `None` : dans ce cas tous les sommets ont été supprimés et il n'y a rien à colorier.

c) Pour toute liste $u$ constituée des éléments $u_0, u_1, u_2 \ldots$ et pour toute fonction $f$, la liste $[f(u_0), f(u_1), f(u_2), \ldots]$ s'écrit très simplement en python : `[f(x) for x in u]` — en remplaçant les crochets par des accolades on obtient un ensemble au lieu d'une liste. On pourra utiliser cette construction pour calculer la liste (ou l'ensemble) des couleurs des voisins d'un sommet.

   Pour écrire la fonction `couleurLibre(s:sommet)` il reste à parcourir la liste des couleurs disponibles (voir annexe ci-dessous) et à retourner la première qui ne figure pas parmi les couleurs des voisins de $s$ ; si ce n'est pas possible (toutes les couleurs sont prises) retourner `'white'`.

d) Utiliser le conseil précédent pour calculer le nombre de couleurs employées par la fonction `colorierGraphe` — la fonction `len(E)` compte le nombre d'éléments d'un *ensemble* $E$.

Tester la fonction `colorierGraphe` sur des grilles, sur le graphe de Petersen (voir exercice 15.1.4), et sur les graphes planaires réguliers (tétraèdre, cube, octaèdre, dodécaèdre et icosaèdre). Utiliser la fonction `melange` comme dans l'exercice 11.6.1 pour que la fonction `sommetNonMarqueMinimal` retourne un sommet aléatoire parmi ceux de degré minimal, lorsqu'il en existe plusieurs : on verra alors que le nombre de couleurs employées pour colorier certains graphes varie suivant l'ordre dans lequel on choisit les sommets de degré minimal.