Introduction à l'informatique
Graphes

Benjamin Monmege, modifié par Thomas Schatz

2021/2022

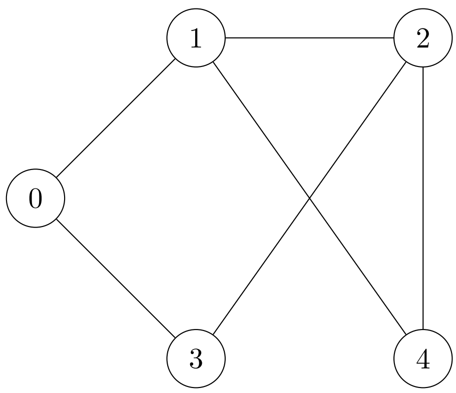
Nous savons désormais mieux comment on (ou une machine) peut calculer avec des algorithmes, sur des ensembles d'objects stockés dans des tableaux. En particulier, on a vu comment trier des tableaux. Nous allons considérer ici un autre problème : le calcul du plus court chemin dans Google Maps : nous allons voir qu'on peut abstraire le problème à l'aide d'un graphe avec des nœuds représentant les intersections de rue et des arcs reliant ces nœuds avec la durée en minutes du trajet correspondant.

1 Graphes

Définissons plus formellement ce qu'est un graphe, pour pouvoir mieux raisonner dessus.

Définition 1. Un graphe est la donnée d'un ensemble fini S de *sommets* (ou nœuds) et d'un ensemble fini A de paires de sommets qu'on appelle *arêtes* : si u et v sont deux sommets, l'arête $\{u, v\}$ représente le fait que les sommets u et v sont reliés dans le graphe. On note parfois $G = (S, A)$ le graphe.

Voici un exemple de graphe pour lequel l'ensemble de sommets est $S = \{0, 1, 2, 3, 4\}$ et l'ensemble d'arêtes est $A = \{\{0, 1\}, \{0, 3\}, \{1, 2\}, \{1, 4\}, \{2, 4\}, \{2, 3\}\}$:



Un *chemin* (on dit parfois aussi *chaîne*) dans un tel graphe $G = (S, A)$ est une suite de sommets $s_1, s_2, s_3, \ldots, s_{n-1}, s_n$ qui sont reliés par une arête, c'est-à-dire tel que $\{s_1, s_2\} \in A$, $\{s_2, s_3\} \in A$, …, et $\{s_{n-1}, s_n\} \in A$. La longueur d'un chemin est le nombre d'arêtes qu'il emprunte. Par exemple $0, 1, 2, 4, 2, 3$ est un chemin de longueur 5 dans le graphe précédent.

Un graphe non orienté permet, par exemple, de représenter les relations de connaissance ou d'amitié dans un réseau social (de telles relations sont généralement symétriques : Marc

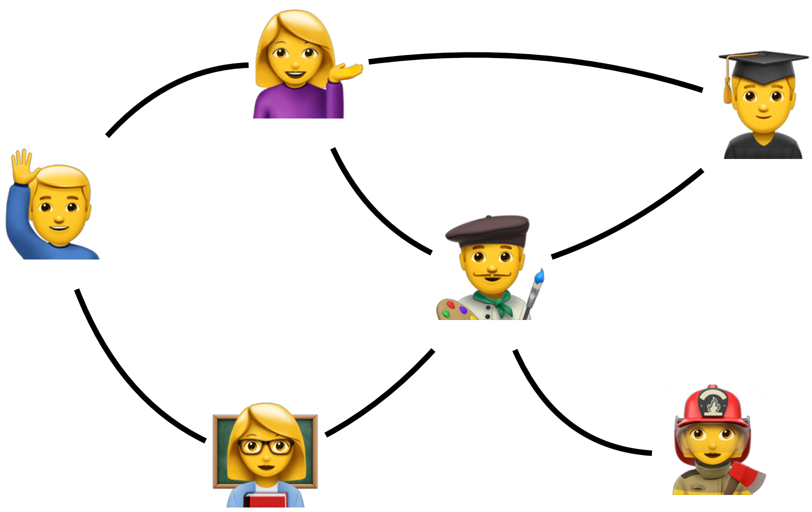


FIGURE 1 – Aperçu du graphe d'un réseau social

est ami avec Sarah si et seulement si Sarah est amie avec Marc...). La FIGURE 1 donne un petit aperçu d'un tel réseau social : les sommets sont les utilisateurs du réseau social et les arêtes représentent les liens de connaissance ou d'amitié.

Considérons désormais le plan d'une ville : il y a des routes qui se croisent à des intersections. On peut donc sur-imprimer au-dessus du plan un graphe dont les sommets sont les intersections et les arêtes sont les morceaux de route sans intersection. On a représenté une partie d'un tel graphe en FIGURE 2.

Trouver un chemin pour aller d'un point A à un point B dans la ville, revient donc à trouver un chemin du sommet A au sommet B dans le graphe. Si cela fonctionne parfaitement pour trouver un chemin pour un piéton, c'est nettement moins intéressant pour une voiture, puisqu'on n'a pas l'information des sens interdits ! Il nous faut donc ajouter cette information dans les graphes, qu'on appelle ainsi *orientés*. Un exemple de graphe orienté pour le plan de Marseille est donné en FIGURE 3.

Définition 2. Un graphe orienté est la donnée d'un ensemble fini S de *sommets* (ou nœuds) et d'un ensemble fini A de couples de sommets qu'on appelle *arcs* : si u et v sont deux sommets, l'arc (u, v) représente le fait qu'il existe un arc partant du sommet u et allant au sommet v. On note parfois $G = (S, A)$ le graphe.

Notez qu'on utilise la notation $\{u, v\}$ pour une paire de sommets (différents), une arête dans un graphe non orienté, alors qu'on utilise la notation (u, v) pour un couple de sommets (potentiellement le même sommet deux fois), un arc dans un graphe orienté. En particulier, si 1 et 2 sont deux sommets du graphe, les arêtes $\{1, 2\}$ et $\{2, 1\}$ sont les mêmes, alors que les arcs $(1, 2)$ et $(2, 1)$ sont différents. Pour les rues dans un plan, une rue à sens unique sera représentée par un arc dans un seul sens, alors qu'une rue à double-sens est représentée par deux arcs, un pour chaque sens.

Voici un exemple de graphe orienté pour lequel l'ensemble de sommets est $S = \{0, 1, 2, 3, 4\}$ et l'ensemble d'arcs (qu'on représente par des liens avec des flèches) est $A = \{(0, 1), (1, 0), (1, 2), (1, 4), (2, 4), (3, 0), (3, 2), (3, 3)\}$:

FIGURE 2 – Graphe sur-imprimé sur un morceau du plan de Marseille

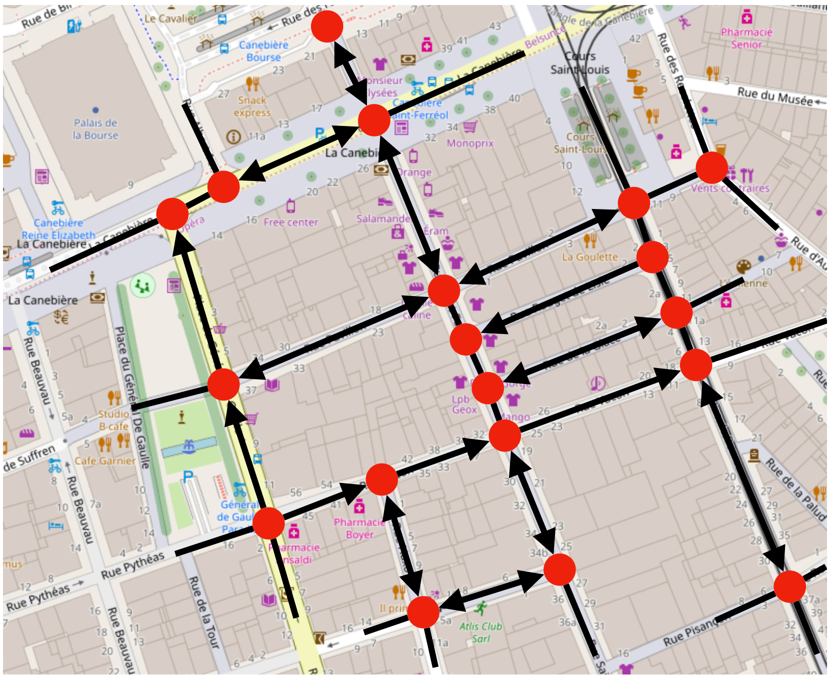
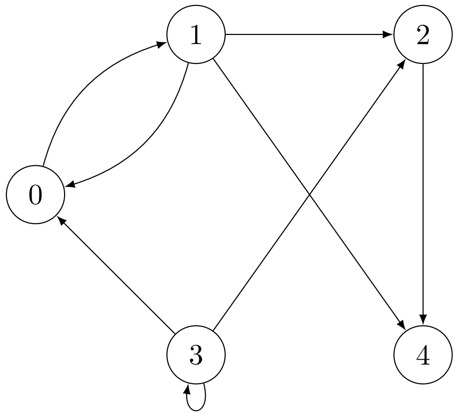


FIGURE 3 – Graphe orienté sur-imprimé sur un morceau du plan de Marseille



Un *chemin* dans un tel graphe orienté $G = (S, A)$ est une suite de sommets $s_1, s_2, s_3, \ldots, s_{n-1}, s_n$ qui sont reliés par un arc, c'est-à-dire tel que $(s_1, s_2) \in A$, $(s_2, s_3) \in A$, ..., et $(s_{n-1}, s_n) \in A$. La longueur d'un chemin est le nombre d'arcs qu'il emprunte. Par exemple $3, 3, 0, 1, 4$ est un chemin de longueur 4 dans le graphe orienté précédent.

2 Codage d'un graphe

Pour pouvoir calculer sur un graphe, il faut savoir comment on va coder un graphe. La façon la plus simple (il existe d'autres façons qu'on n'étudiera pas dans ce cours) consiste à stocker le graphe à l'aide d'une *matrice d'adjacence*. Il s'agit d'un tableau bidimensionnel M dont les lignes et les colonnes sont indexées par les sommets du graphe et tel que la case $M[u, v]$ en ligne u et en colonne v vaut 1 dès lors qu'il existe un arc (u, v) ou une arête $\{u, v\}$ dans le graphe, et vaut 0 sinon. Voici les matrices d'adjacence du graphe non orienté (à gauche) et orienté (à droite) présentés plus tôt :

$$M_1 = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{pmatrix} \qquad M_2 = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Par exemple, on a $M_1[2, 3] = 1$ signifiant qu'il existe une arête reliant les sommets 2 et 3 dans le graphe non orienté, et $M_2[2, 1] = 0$ signifiant qu'il n'y a pas d'arc allant de 2 à 1 dans le graphe orienté.

3 Parcours de graphe et algorithme de Dijkstra dans les graphes pondérés

On considère un graphe orienté $G = (S, A)$ avec un poids positif associé à chaque arête représentant par exemple la distance entre les deux sommets associés à l'arête. L'algorithme suivant permet de trouver le plus court chemin entre un sommet de départ s_{deb} et chacun des autres sommets du graphe (la longueur d'un chemin est la somme des poids des arêtes traversées en suivant le chemin).

```
Entrées : G = (S, A) un graphe avec une pondération positive poids des arcs, s_deb un sommet de S

P := ∅
d[a] := +∞ pour chaque sommet a
d[s_deb] = 0
Tant qu'il existe un sommet hors de P
    Choisir un sommet a hors de P de plus petite distance d[a]
    Mettre a dans P
    Pour chaque sommet b hors de  P voisin de a
       Si d[b] > d[a] + poids(a, b)
           d[b] = d[a] + poids(a, b)
           prédécesseur[b] := a
    Fin Pour
Fin Tant Que
```

FIGURE 4 – Algorithme de Dijskstra pour la découverte des plus courts chemins.

Commencez à réfléchir aux questions suivantes. Cet algorithme termine-t-il toujours ? Comment résoud-il le problème du plus court chemin ? Quelle est sa complexité ?

4 Graphes eulériens : le problème des sept ponts de Königsberg

On peut trouver de nombreux autres problèmes liés aux graphes dans la vie courante. Considérons dans un premier temps un problème concret que Leonhard Euler, un mathématicien du XVIIIème siècle, alors résident à Saint-Petersbourg, s'est posé en visitant la ville de Königsberg (qui s'appelle Kaliningrad de nos jours). Cette ville est séparée en plusieurs parties par la rivière Pregel au-dessus de laquelle sept ponts passent, comme le montre le plan en FIGURE 5.

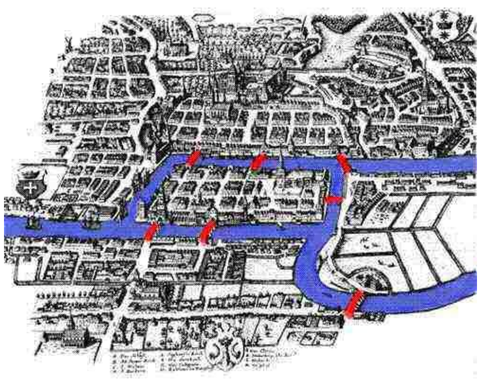


FIGURE 5 – Les sept ponts de Königsberg

Un guide touristique se pose alors la question, pour attirer une foule de touristes toujours plus grande dans ses visites, de savoir s'il est possible de faire visiter la ville en faisant un circuit passant une fois par chaque pont, mais sans passer deux fois par le même pont. On veut donc partir d'un endroit dans la ville, faire un tour qui passe par chaque pont une et une seule fois, puis être revenu au point de départ.

On voit bien que la carte précédente recèle bien trop de détails qu'on peut ignorer pour

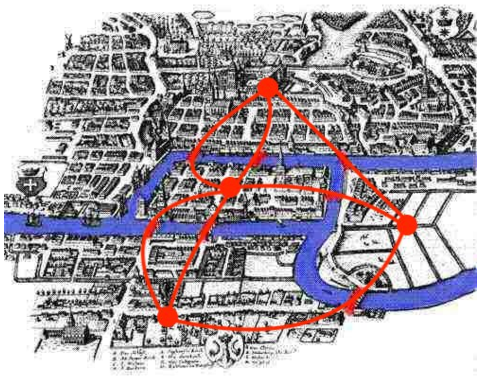
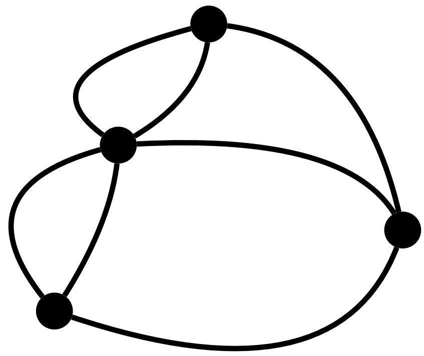


FIGURE 6 – Le graphe sous-jacent au problème des sept ponts de Königsberg

répondre à la question. Il suffit de ne garder que les ponts et la façon dont ils sont reliés entre eux, par les rives et les îles de Königsberg, comme représenté en FIGURE 6.

Si on ne garde que le graphe représenté en rouge, on arrive à la modélisation suivante :



Notez que ce graphe est un peu particulier. C'est un graphe non orienté ayant quatre sommets, mais il y a plusieurs arêtes reliant la même paire de sommets : on appelle souvent ce genre de graphe des multi-graphes, signifiant qu'on s'autorise à mettre plusieurs arêtes plutôt qu'une seule entre certaines paires de sommets. Dans un tel graphe non orienté, on appelle *cycle eulérien* tout cycle (un chemin qui revient à son point de départ) qui traverse chaque arête du graphe une et une seule fois. Partez donc de n'importe quel sommet, puis essayez de créer un cycle eulérien dans le graphe précédent.

Vraisemblablement, vous n'y arriverez pas : il vous restera toujours une arête au moins non visitée alors que vous serez coincé dans un sommet d'où vous avez déjà visité toutes les arêtes y arrivant... Rassurez-vous, ce n'est en rien de votre faute. Leonhard Euler lui-même s'est fait la même réflexion et a fini par se convaincre que, dans ce graphe, il n'y a pas de cycle eulérien. Il en a même déduit une recette miraculeuse permettant de décider en un instant s'il y a un cycle eulérien ou pas dans un graphe. Pour cela, notons dans un premier temps que la présence d'un cycle visitant toutes les arêtes du graphe n'est possible que si le graphe est en un seul morceau, c'est-à-dire qu'il est possible de relier toute paire de sommets par un chemin (une suite d'arêtes consécutives) : on parle alors de graphes *connexes*. Le graphe de Königsberg est connexe. Sa recette miraculeuse consiste à compter, pour un sommet quelconque, le nombre d'arêtes arrivant dans ce sommet, qu'on appelle le *degré* du sommet. Dans le graphe de Königsberg, trois sommets ont degré 3 et le sommet représentant l'île centrale a degré 5. Euler remarqua que s'il existe un cycle eulérien, alors celui-ci visite toutes les arêtes du graphe (par définition) : pour chaque sommet, on doit donc rentrer autant de fois dedans qu'on en sort. Autrement dit, le degré de chaque sommet doit être pair. Ce qui est intéressant, c'est que cette condition est aussi suffisante :

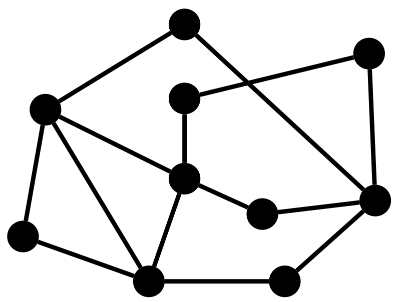
Théorème 1. *Un graphe non orienté connexe admet un cycle eulérien si et seulement si chaque sommet est de degré pair.*

Cette condition permet tout de suite d'affirmer qu'il n'y a donc pas de cycle eulérien dans le graphe de Königsberg, puisque les quatre sommets ont un degré impair. Comment prouver la condition suffisante du théorème d'Euler, à savoir que si chaque sommet d'un graphe connexe a un degré pair, alors il existe un cycle eulérien ? Utilisons une preuve par algorithme, c'est-à-dire qu'on va donner un algorithme qui construit un cycle eulérien dès lors que les sommets sont tous de degré pair : la preuve de terminaison et de correction de l'algorithme donne alors une preuve du théorème d'Euler.

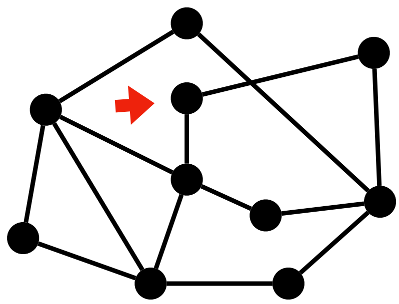
C'est l'algorithme de Hierholzer qui nous permet de construire un cycle eulérien (s'il en existe un). On peut le décrire de manière informelle ainsi :

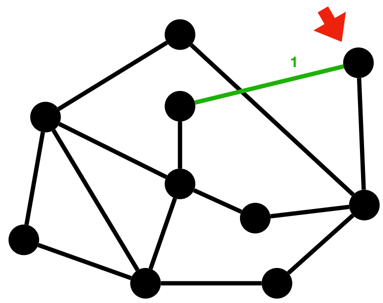
- Choisir n'importe quel sommet initial v
- Suivre un chemin arbitraire d'arêtes jusqu'à retourner à v, obtenant ainsi un cycle c
- **Tant qu**'il y a des sommets u dans le cycle c avec des arêtes qu'on n'a pas encore choisies **faire**
  - Suivre un chemin à partir de u, n'utilisant que des arêtes pas encore choisies, jusqu'à retourner à u, obtenant un cycle c'
  - Prolonger le cycle c par c'

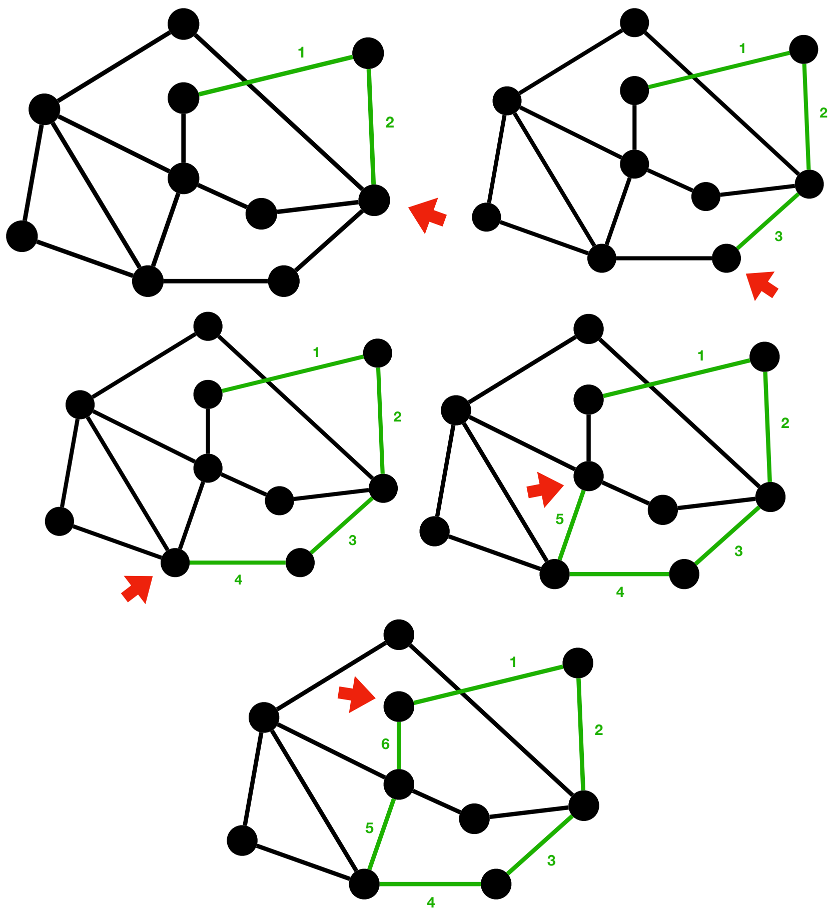
Voici une exécution de l'algorithme sur le graphe suivant :



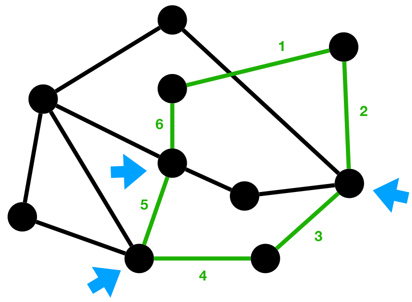
On choisit arbitrairement un sommet initial (marqué par la flèche rouge), puis on suit un chemin arbitraire d'arêtes jusqu'à être retourné au sommet initial :



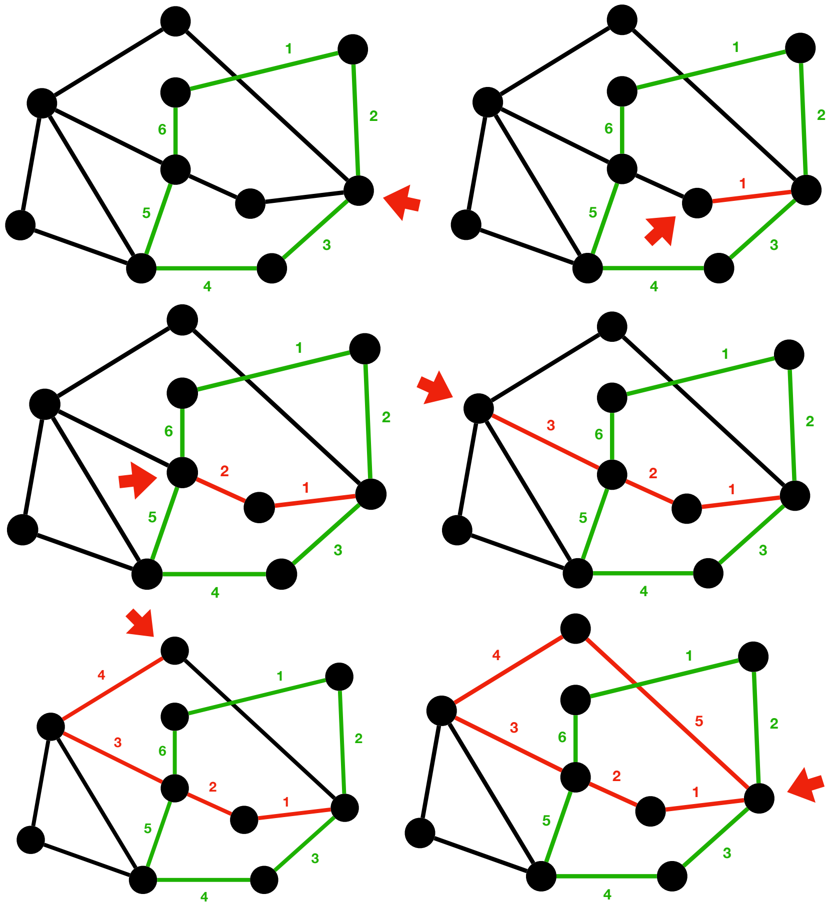




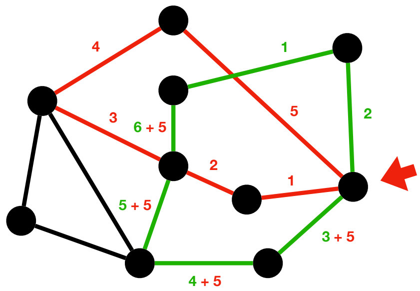
On a obtenu un cycle c, mais ce n'est pas encore un cycle eulérien puisque certaines arêtes ne sont pas visitées. Dans ce cycle, trois sommets (marqués par les flèches bleues ci-dessous) ont des arêtes qu'on n'a pas encore choisies :



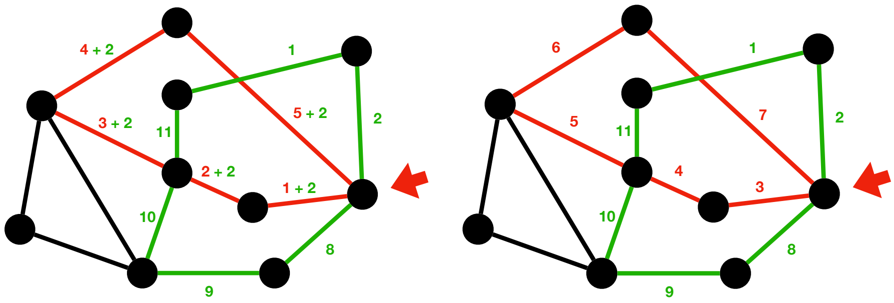
On choisit l'un de ces sommets, puis on recherche à nouveau un cycle c' n'empruntant aucune des arêtes préalablement choisies :



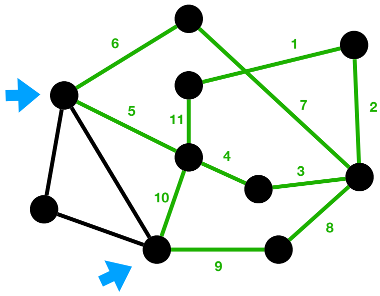
Il reste à insérer le cycle c' au sein du cycle c, c'est-à-dire qu'après avoir visité les deux premières arêtes de c, on se met en pause pour visiter le cycle c' entièrement, avant de terminer avec les quatre dernières arêtes de c. Pour faire cela, il suffit d'ajouter au numéro des quatre dernières arêtes de c la longueur de c' :



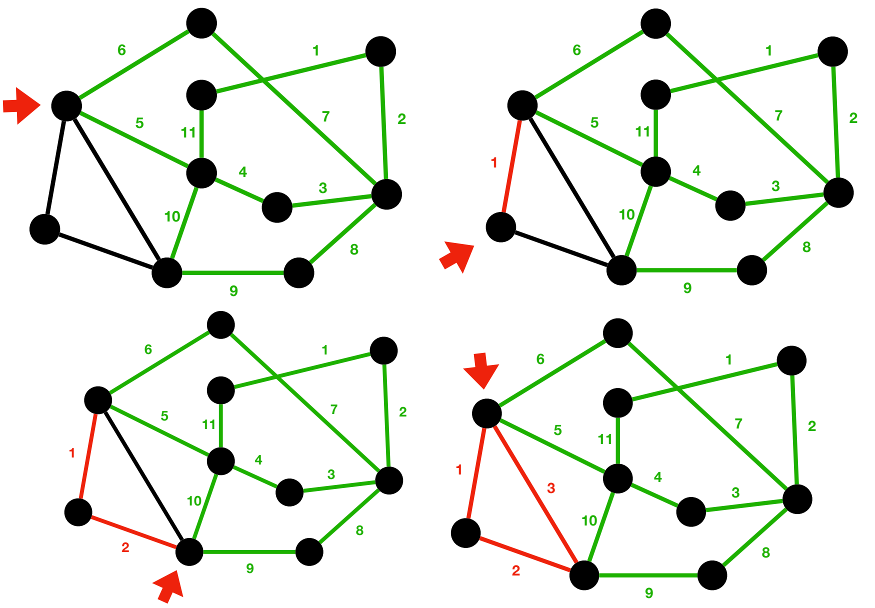
puis de décaler les numéros des arêtes du cycle c de deux pour l'insérer au bon endroit :

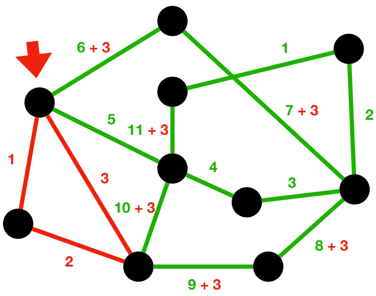


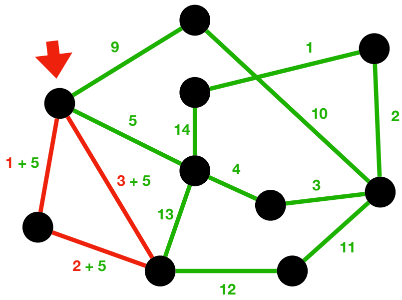
On obtient donc un cycle un peu plus long qui ne visite pas deux fois la même arête. Ce n'est toujours pas un cycle eulérien : deux sommets dans le cycle ont encore des arêtes non visitées



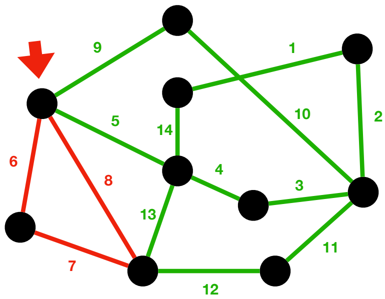
On en choisit un puis on recommence la même recherche de cycle, puis l'insertion du petit cycle dans le grand :







On obtient finalement le cycle eulérien suivant :



Pourquoi cet algorithme termine-t-il et est-il correct si le graphe vérifie la condition de connexité et sur les degrés de ses sommets ? Tout d'abord, remarquons qu'on ne peut pas rester bloqué à un moment dans la recherche d'un nouveau cycle : en effet, du fait de la parité du degré des sommets, si on entre dans un sommet, on peut toujours en sortir. La recherche du premier cycle finit tôt ou tard par terminer puisqu'on finit nécessairement par retourner dans le sommet initial v : l'ensemble des arêtes est fini et le sommet v a un nombre impair d'arêtes inexplorées une fois qu'on en est parti (donc il reste au moins une arête pour y entrer). Ces arguments sont encore vrai pour la recherche du cycle autour de u, ce qui fait que l'algorithme de Hierholzer termine. Il produit bien un cycle qui visite toutes les arêtes du graphe, sinon, par connexité, au moins un des sommets devraient avoir encore une arête non visitée. Finalement, il ne peut pas visiter deux fois la même arête, par construction : il produit donc bien un cycle eulérien. Cela prouve donc le théorème d'Euler.

5 Coloration de graphes et des cartes

Considérons un autre problème lié aux graphes. Imaginons la tâche d'un géographe qui souhaite colorier les pays d'une carte du monde avec le moins de couleurs possibles, tout en garantissant toujours que deux pays partageant une frontière terrestre n'ont pas la même couleur. On peut par exemple produire la carte de la Figure 7, utilisant quatre couleurs. Est-ce un hasard ? En fait, non, c'est toujours possible comme le dit le théorème des quatre couleurs :

Théorème 2. *On peut colorer n'importe quelle carte avec un maximum de quatre couleurs de sorte que les zones adjacentes reçoivent toujours deux couleurs distinctes.*

Ce théorème très simple a posé de fortes résistances à la communauté scientifique. Conjecturé dès le milieu du XIXème siècle, plusieurs preuves erronées ont été proposées, jusqu'à ce

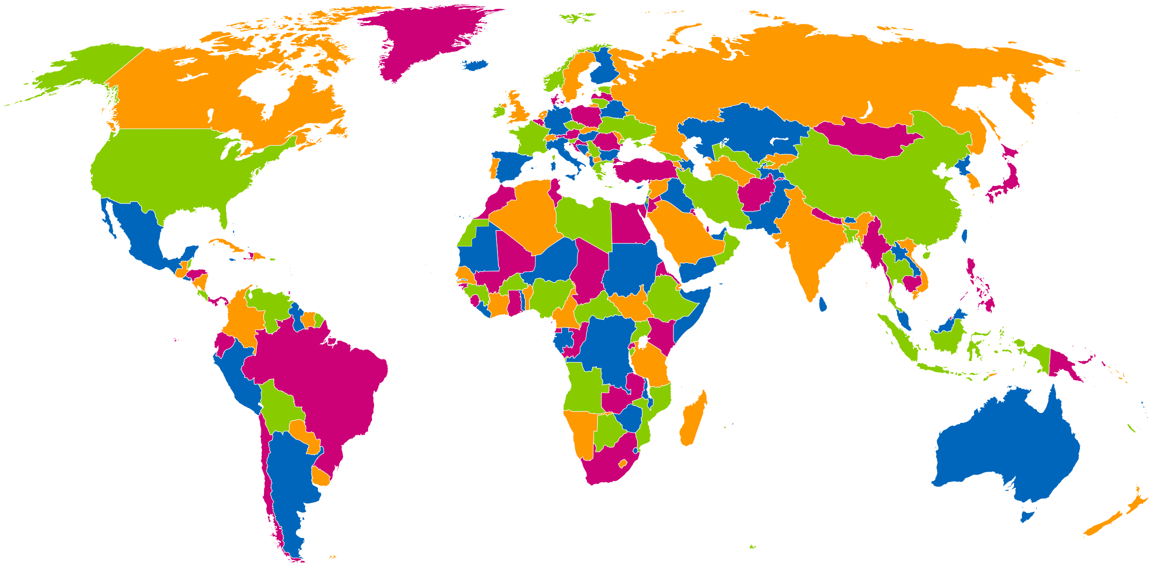
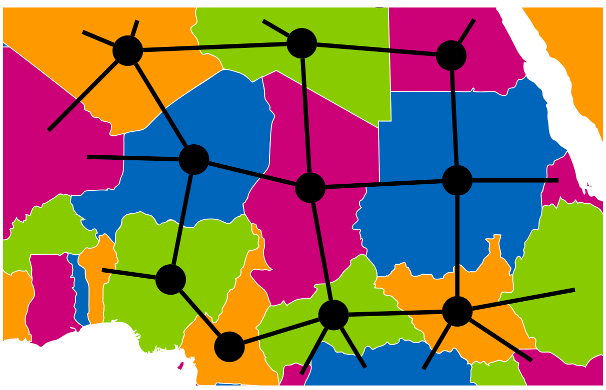


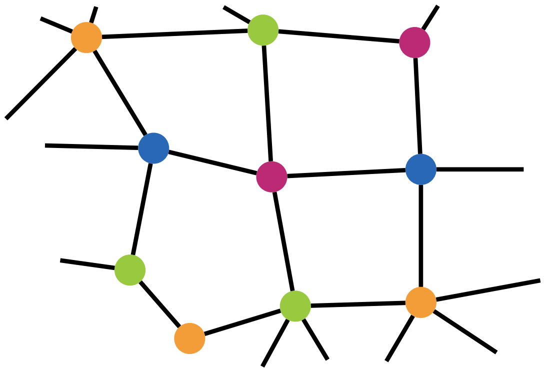
FIGURE 7 – Mappemonde colorée avec quatre couleurs

que, dans les années 1960 et 1970, des chercheurs utilisent des ordinateurs pour les aider à produire une preuve. En effet, on peut prouver le théorème avec des arguments relativement simples dès lors que la carte à colorier contient suffisamment de pays : mais cela laisse de nombreux « petits » graphes à étudier qu'il est très pénible, sinon impossible à tous faire à la main. L'utilisation de l'ordinateur s'avère donc indispensable mais a alors partagé la communauté scientifique pour savoir si cela pouvait effectivement encore s'appeler une preuve. Ce qui est sûr, c'est que le problème de prouver le théorème se déplace (comme dans le cas de l'algorithme de Hierholzer pour le théorème d'Euler) sur la preuve de correction de l'algorithme qui sert à vérifier les « petits » graphes. À l'heure actuelle, on ne connait toujours pas de preuve entièrement à la main de ce théorème pourtant simple à énoncer.

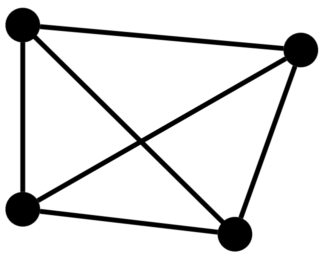
Mais quel rapport avec les graphes ? Si on zoome un peu sur la carte, on peut modéliser le problème avec un graphe non orienté dont les sommets seront les pays à colorier et les arêtes sont les frontières reliant ces pays :



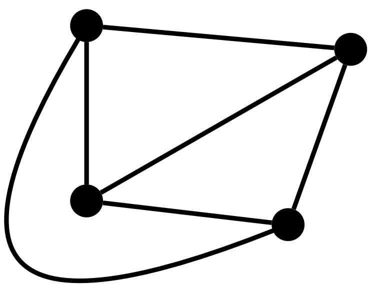
Colorier la carte revient donc à colorier les sommets du graphe, de sorte que deux sommets reliés par une arête ne sont pas coloriés avec la même couleur :



On appelle graphe *planaire* tout graphe qu'on peut obtenir par cette méthode à partir d'une carte quelconque : c'est donc un graphe qu'on peut dessiner sur un plan (une feuille de papier) sans qu'aucune arête n'en croise une autre. Attention, certains graphes ne semblent pas planaires, comme celui-ci...



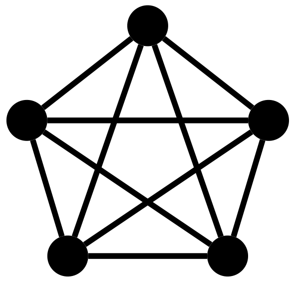
mais le sont en fait dès lors qu'on tord un peu certaines de leurs arêtes :



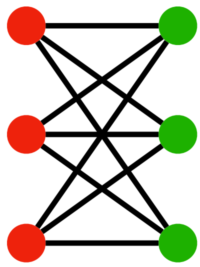
Le théorème des quatre couleurs peut donc s'énoncer de la façon suivante :

Théorème 3. *On peut colorer les sommets de tout graphe non orienté planaire avec un maximum de quatre couleurs de sorte que les sommets adjacents (c'est-à-dire reliés par une arête) reçoivent toujours deux couleurs distinctes.*

Ce théorème permet de se convaincre que certains graphes ne sont pas planaires. Il suffit pour cela de montrer qu'ils ne sont pas coloriables avec quatre couleurs. C'est par exemple le cas du graphe complet (c'est-à-dire tel que toute paire de sommets est une arête) à cinq sommets (ou plus) :



On ne peut pas le colorier avec quatre couleurs puisque chaque sommet a quatre sommets adjacents qui sont eux-même reliés entre eux. Par contre, noter que le théorème des quatre couleurs ne donne pas une condition nécessaire et suffisante pour être un graphe planaire. Le graphe suivant n'est pas planaire, mais il peut être colorié avec deux couleurs :



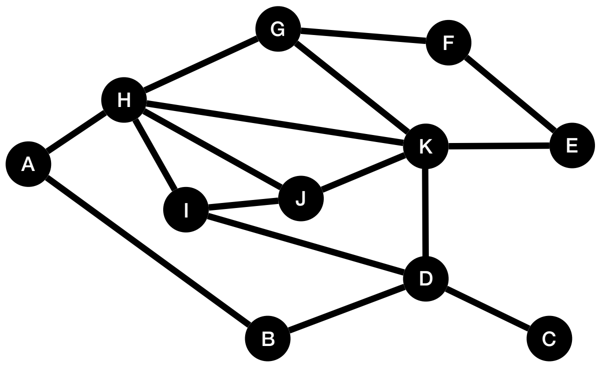
Le théorème des quatre couleurs a une grosse lacune : il dit qu'il est possible de colorer un graphe planaire avec quatre couleurs, mais ne donne pas de méthode pour faire cela. Comment donc colorer effectivement un graphe planaire, ou une carte de géographie, à partir de là ?

Une première méthode naïve pourrait consister à essayer toutes les possibilités de coloriage. Dans le pire des cas, cela demande à essayer toutes les possibilités de coloriage des sommets du graphe avec quatre couleurs. Un coloriage est une fonction de l'ensemble S des sommets dans un ensemble de couleurs $\{0, 1, 2, 3\}$ à quatre éléments : ces fonctions sont au nombre de $4^{|S|}$, qui est exponentiel en fonction du nombre de sommets. Il n'est donc clairement pas envisageable d'essayer toutes les possibilités, dès lors que le graphe est un peu gros.

À la place, on étudie l'algorithme de Welsh-Powell, qu'on peut écrire de la manière suivante :

- Trier les sommets du graphe par ordre de degré décroissant
- couleur $\leftarrow$ 0 (*couleur initiale*)
- **Tant qu**'il y a encore des sommets non colorés **faire**
  - Parcourir la liste triée des sommets et colorer en *couleur* les sommets non colorés qui ne sont pas connectés à d'autres sommets de la même couleur
  - couleur $\leftarrow$ couleur + 1 (*choisir une nouvelle couleur*)

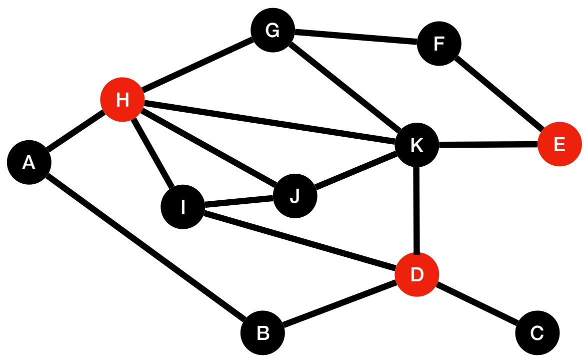
L'idée est donc très simple : on colorie les uns après les autres les sommets du graphe, en le traitant dans l'ordre décroissant de leur degré, et on essaie d'attribuer à un maximum de sommets la couleur 0, puis la couleur 1, puis la couleur 2, etc. Considérons par exemple le graphe suivant dont les sommets sont les lettres de A à K :



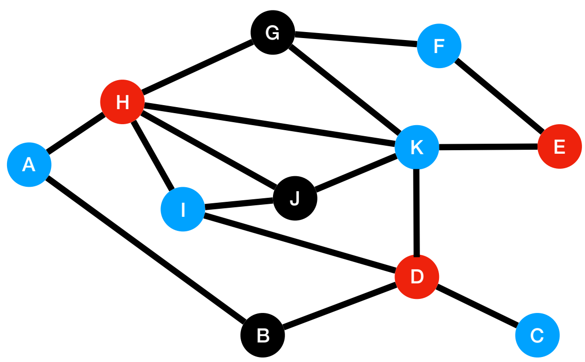
On commence par trier les sommets par ordre décroissant de degrés :

H	K	D	G	I	J	A	B	E	F	C
5	5	4	3	3	3	2	2	2	2	1

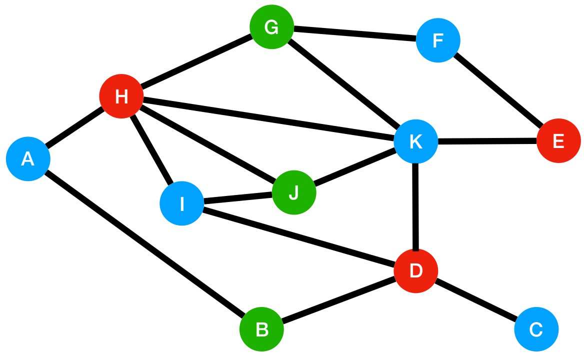
On commence par vouloir attribuer la couleur 0, disons rouge. On parcourt ainsi la liste précédente des sommets et on colorie le sommet en rouge, dès lors qu'aucun de ses voisins n'a déjà cette couleur. On peut donc colorer les sommets H, D et E :



On passe alors à la couleur suivante, disons bleu. On parcourt la liste des sommets non encore colorés, et on colorie le sommet en bleu dès lors qu'aucun de ses voisins n'a déjà cette couleur. On peut alors colorer les sommets K, I, A, F et C :

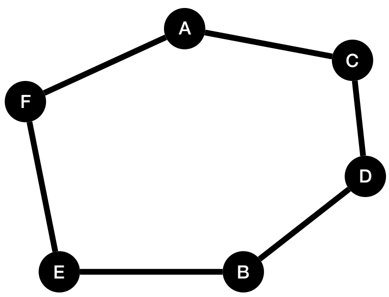


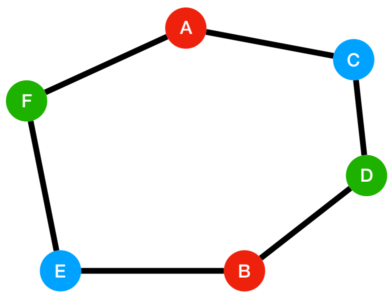
On fait de même avec la couleur suivante, le vert par exemple, pour colorer les sommets restants G, J et B :



On a donc réussi à colorer ce graphe planaire avec trois couleurs seulement. Notons qu'il n'est pas possible de faire mieux, c'est-à-dire de n'utiliser que deux couleurs seulement. En effet, ce graphe possède un *triangle*, c'est-à-dire trois sommets reliés les uns aux autres, par exemple les sommets H, I et J : un tel triangle nécessite trois couleurs puisque chaque sommet est relié à deux autres sommets reliés entre eux. Sur ce graphe, l'algorithme de Welsh-Powell renvoie donc une coloration *optimale*, c'est-à-dire avec le nombre minimal de couleurs possible.

Ce n'est malheureusement pas toujours le cas. Par exemple, sur le graphe suivant, où tous les sommets ont degré 2, si on suit l'ordre alphabétique de traitement des sommets, on arrive à la coloration à droite avec trois couleurs :





En fait, ce graphe ne nécessite que deux couleurs, puisqu'on peut colorer les sommets A, D et E d'une même couleur, puis B, C et F d'une autre même couleur.