Théorie des graphes

Marc Abboud
Séance Rennes et Maths

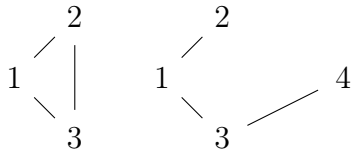
21 Novembre 2021

1 Combinatoire sur les graphes

Définition 1.1. Formellement, un graphe est la donnée d'un ensemble de sommets S et d'un ensemble d'arêtes $A \subseteq S \times S$. Si $i, j \in S$, on dit qu'ils sont voisins si $(i, j) \in S$.

Dans la suite sauf si le contraire est mentionné, on supposera que nos graphes ne sont pas orientés, c'est à dire pour tout $i, j \in S, (i, j) \in A \Leftrightarrow (j, i) \in A$ et qu'ils sont simples c'est à dire que tout arête apparait au plus une fois et qu'ils sont sans boucles, c'est à dire que pour tout $s \in S, (s, s) \notin A$. La taille d'un graphe fini sera le cardinal de S.

Exemple 1.2. Un graphe de taille 3 et un de taille 4.



Exercice 1. Donner le nombre de graphes de taille n, dessiner tous les graphes de taille 3.

Définition 1.3. Soit G un graphe et v un sommet de G, le *degré* de v est le nombre d'arêtes dont une des extrémités est v. On le note $d(v)$.

Définition 1.4 (Sous-graphe)**.** Soit $G = (S, A)$ un graphe, un sous-graphe $G' = (S', A')$ est un *sous-graphe* de G si $S' \subseteq S, A' \subseteq A$.

Exemple 1.5. Cet exemple nous sera utile dans la suite, si G est un graphe et v un sommet, on note $G \setminus \{v\}$ le sous-graphe de G obtenu en enlevant le sommet v et toutes les arêtes de G dont une des extrémités est v.

Exercice 2. On note K_n le graphe de taille n dont tous les sommets sont reliés entre eux. Montrer que tout graphe de taille $\leq n$ est un sous-graphe de K_n.

Exercice 3. Montrer que pour tout graphe $G = (S, A)$, on a

$$\sum_{v \in S} d(v) = 2\,|A|$$

Définition 1.6. Soit $G = (S, A)$ un graphe, un chemin dans G est la donnée de sommets $v_1, ..., v_k$ de sorte que pour tout $i = 1, ..., k-1$, $(v_i, v_{i+1}) \in A$. Un chemin est un cycle si $v_1 = v_k$.

Définition 1.7. Un graphe G est dit *connexe* si pour tous sommets u, v de G, il existe un chemin reliant u et v.

Définition 1.8. Soit G un graphe et v un sommet de ce graphe. On appelle la *composante connexe de* v l'ensemble défini par

$$C(v) = \{v' \in S(G) : \text{il existe un chemin de } v \text{ vers } v'\}$$

Exercice 4.

1. Montrer que $v \in C(v)$ et que $v \in C(v') \Leftrightarrow v' \in C(v)$.
2. Montrer que s'il y a un chemin de v vers v' et un chemin de v' vers w alors il y a un chemin de v vers w.
3. En déduire que pour deux sommets v, v' de G, on a

$$C(v) \cap C(v') \neq \emptyset \Leftrightarrow C(v) = C(v').$$

   $C(v)$ est une des *composantes connexes* de G.
4. Montrer que G est réunion disjointe de ses composantes connexes.

Exercice 5. En procédant par récurrence sur le nombre de sommets, montrer qu'un graphe connexe de taille n a au moins $n-1$ arêtes.

1.1 Les arbres

Définition 1.9. Un *arbre* est un graphe connexe, sans cycle.

Exercice 6. On veut montrer qu'un arbre à n sommets a $n-1$ arêtes.

1. On procède par récurrence sur n. Montrer le résultat pour $n = 1$.
2. On suppose le résultat vrai pour un arbre à $n-1$ sommets et on considère G un arbre à n sommets. Montrer que G possède un sommet de degré 1 que l'on notera v.
3. Conclure en considérant le sous-graphe $G \setminus \{v\}$.

Proposition 1.10. *Soit G un graphe à n sommets. Les assertions suivantes sont équivalentes.*

(i) *G est un arbre.*

(ii) *G ne contient pas de cycle et a $n-1$ arêtes.*

(iii) *G est connexe et possède $n-1$ arêtes.*

(iv) *G est connexe et chaque arête est un pont. (C'est à dire que pour toute arête e, $G - e$ n'est plus connexe).*

(v) *Deux sommets quelconques de A sont reliés par un unique chemin.*

(vi) A ne contient pas de cycle mais l'ajout quelconque d'une arête à G crée un cycle.

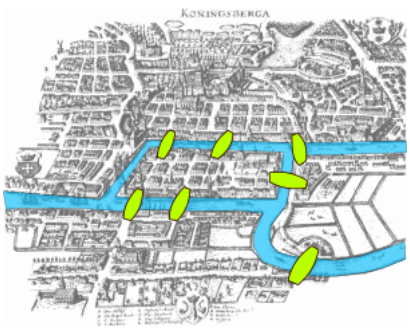
Exercice 7. Montrer la proposition en montrant les équivalences suivantes : $(i) \Rightarrow (ii) \Rightarrow (iii) \Rightarrow (iv) \Rightarrow (v) \Rightarrow (vi) \Rightarrow (i)$.

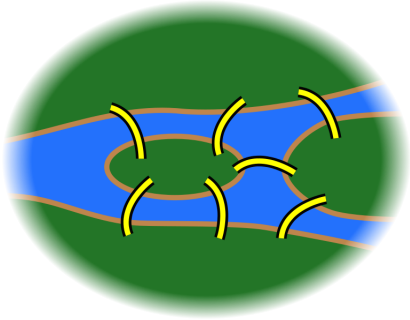
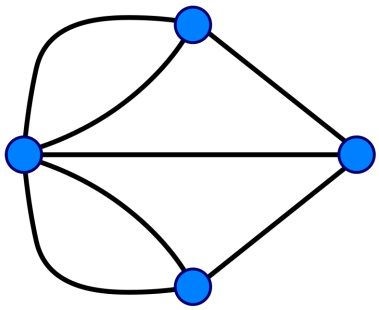
Les arbres sont une structure de données très importante et très utilisée en pratique mais nous n'aurons pas le temps de voir cela pendant la séance.

1.2 Circuit Hamiltonien et Eulérien

1.2.1 Circuit Eulérien

Lorsqu'on fait un cours sur la théorie des graphes, on se doit de parler des ponts de Königsberg. La ville de Königsberg (aujourd'hui Kaliningrad) est construite autour de deux îles situées sur le Pregel et reliées entre elles par un pont. Six autres ponts relient les rives de la rivière à l'une ou l'autre des deux îles. La question que se sont posées les habitants est : Est-il possible de faire une promenade dans les rues de Königsberg en passant une et une seule fois par chacun des ponts de la ville et en revenant à son point de départ ? On peut répondre à cette question en passant par la théorie des graphes comme le montre le dessin ci-dessous.



On définit la notion de circuit/chemin Eulérien.

Définition 1.11. Soit G un graphe, on dit qu'un chemin (resp.circuit) c est eulérien s'il passe une et une seule fois par chaque arête du graphe G. Un graphe est eulérien s'il possède un circuit eulérien.

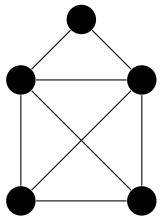
Exercice 8. Montrer que si G possède un circuit eulérien alors nécessairement tous les sommets de G sont de degré pair. Montrer que si G possède un chemin eulérien, alors tous les sommets de G sont de degré pair sauf le premier et dernier sommet du chemin qui ont un degré impair s'ils sont différents.

La réciproque est en fait vraie.

Théorème 1.12. *G possède un circuit eulérien si et seulement si il est connexe et tous ses sommets sont de degré pair. G possède un chemin eulérien qui n'est pas un cycle si et seulement si tous les sommets de G sont de degré pair sauf exactement deux.*

Corollaire 1.13. *Le problème des ponts de Königsberg n'a pas de solutions.*

Exercice 9. Est-ce que le graphe suivant possède un circuit eulérien ?



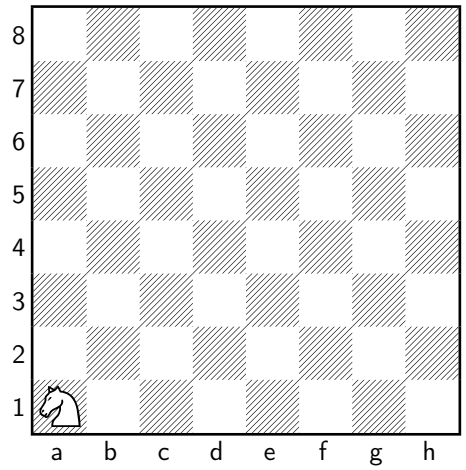
1.2.2 Circuit Hamiltonien

On pourrait maintenant se poser la question analogue suivante. Étant donné un graphe, est-il possible de trouver un chemin passant une et une seule fois par chacun des sommets du graphe ?

Définition 1.14. Soit G un graphe un chemin est *hamiltonien* s'il passe une et une seule fois par chaque sommet du graphe.

Cette question est en fait beaucoup plus dure et il n'y a pas à ce jour de critère simple et général pour montrer qu'un graphe possède un chemin hamiltonien.

Exemple 1.15. On peut se demander si sur un échiquier, un cavalier peut parcourir toutes les cases une et une seule fois en partant de la case a1.



Une solution est a1 c2 e3 g4 h6 g8 e7 c8 a7 b5 d6 f7 h8 g6 f8 d7 b8 a6 c7 e8 f6 h7 g5 e6 g7 f5 d4 c6 d8 b7 a5 b3 c5 a4 b2 d1 c3 b1 a3 c4 e5 d3 e1 g2 h4 f3 h2 f1 d2 e4 f2 h1 g3 h5 f4 h3 g1 e2 c1 a2 b4 d5 b6 a8.

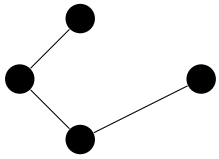
Ce problème peut être représenté par le graphe où les sommets sont les cases de l'échiquier et les arêtes relient deux sommets si un cavalier peut aller de l'un vers l'autre. La question est alors de savoir si ce graphe possède un chemin hamiltonien. La réponse est en fait positive mais le seul moyen que l'on a pour le prouver est d'exhiber un tel chemin et cela se fait par des algorithmes de recherche dans les graphes que nous allons traiter en deuxième partie.

Théorème 1.16. *Soit G un graphe connexe à $n \geq 3$ sommets, si chaque sommet de G a un degré $\geq n/2$ alors G possède un chemin hamiltonien.*

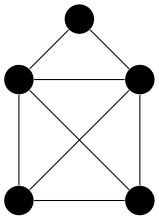
1.3 La formule d'Euler pour les graphes planaires

Définition 1.17. Un graphe planaire est un graphe que l'on peut représenter dans le plan affine de sorte qu'aucune des arêtes du graphe ne se croise sauf en les sommets.

Exemple 1.18.



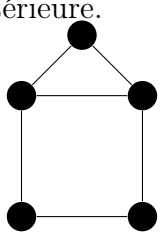
Ceci est un graphe planaire.



Pas celui-ci.

Définition 1.19. Soit G un graphe planaire, considérons le plan affine $\mathbf{R}^2$ privé du graphe. Cela donne plusieurs "morceaux" du plan que l'on appelle les *faces* de G. On note F l'ensemble des faces d'un graphe.

Exemple 1.20. Le graphe suivant possède 3 faces, celle du carré, celle du triangle et la face extérieure.



Exercice 10. Montrer qu'un graphe est sans cycle si et seulement si il n'a qu'une seule face. (Un bon dessin suffira...)

Théorème 1.21 (Euler)**.** *Soit G un graphe planaire connexe, alors*

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

Exercice 11. Montrer le théorème en faisant une récurrence sur le nombre de faces du graphe.

2 Algorithmes sur les graphes

2.1 Complexité

La notion la plus cruciale lorsque l'on parle d'algorithmes est celle de complexité. Concrètement trouver un algorithme qui répond à un problème est une bonne chose mais si l'algorithme mets plus de 10 ans à donner une réponse son intérêt est assez limité...

Définition 2.1. Étant donné un algorithme, sa complexité est le nombre d'opérations élémentaires que celui-ci effectue pour rendre un résultat. Elle est exprimée en fonction des paramètres des entrées.

Définition 2.2. Soit $f, g : \mathbf{N} \to \mathbf{R}$ deux fonctions, on dit que $f = O(g)$ si à partir d'un certain rang, il existe une constante $c > 0$ telle que $f(n) < c \cdot g(n)$.

On utilise beaucoup cette notion car en pratique on ne s'intéresse pas au nombre exact d'opérations mais on cherche à le borner pour savoir si notre algorithme est efficace ou non.

Exemple 2.3. Supposons avoir une liste non triée de n entiers, alors trouver l'élément minimal se fait naïvement en $O(n)$ opérations (en fait on peut avoir le minimum en $O(\log n)$ opérations avec des algorithmes du type diviser pour régner).

Exemple 2.4. David Saulpic pendant sa séance sur l'algorithmique vous a montré que l'on peut trier une liste de taille n en $O(n \log n)$ opérations.

3 Algorithme de parcours d'un graphe

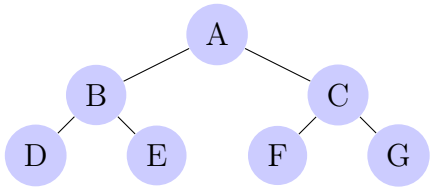
A partir de maintenant, on ne va plus représenter un graphe par un ensemble de sommets S et un ensemble d'arêtes A. On va plutôt se donner un ensemble de sommets S et pour tout sommet $v \in S$, on se donne $V(v) \subseteq S$ l'ensemble des voisins de v. C'est strictement équivalent mais beaucoup plus commode lorsqu'on veut faire des algorithmes sur un graphe. On va étudier deux types de parcours, le parcours en pronfondeur et le parcours en largeur. On dit qu'on représente le graphe par sa *liste d'adjacence*.

3.1 Le parcours en largeur

On l'appelle comme ça car on part d'un sommet v et on lit l'ensemble de ses voisins. Ensuite pour chaque voisin de v on va aussi lire l'ensemble de ses voisins etc... Algorithmiquement, cela pourrait se traduire comme ceci :

Données : Un graphe G avec sa liste d'adjacence et un sommet v
File $= \{v\}$;
Marquer v ;
tant que *File* $\neq \emptyset$ **faire**
  Sortir le premier élément de la file s, marquer s ;
  Afficher s;
  pour t *voisins de* s **faire**
    si t *n'est pas marqué* **alors**
      Mettre t à la fin de la file. Marquer t.
    fin
  fin
fin

Exemple 3.1. Supposons que l'on ait un arbre



alors le parcours en largeur de l'arbre va donner A,B,C,D,E,F,G.

Exercice 12. Montrer que l'algorithme parcourt bien toute la composante connexe du sommet v et qu'il termine. Donner la complexité du parcours en largeur.

Exercice 13. Donner un algorithme pour déterminer si un graphe est connexe en utilisant un parcours en largeur.

3.1.1 Algorithme de Dikjstra

Ici on va s'intéresser au cas plus général de graphe dont les arêtes sont pondérés. C'est à dire qu'on va attribuer un poids positif à chaque arête (Par exemple si on considère le graphe des villes de france, on relie toutes les villes entre elles en pondérant les arêtes par les distances entre chaque ville). On cherche alors le plus court chemin entre deux points dans le graphe. Cet algorithme se base sur un parcours en largeur.

L'algorithme que je décris ici est le plus général et permet de donner le chemin le plus court d'un sommet fixé à tous les sommets du graphe.

Données : Un graphe G avec des arêtes pondérées et sa liste d'adjacence et un sommet v_{deb}
$P = \emptyset$;
$d(s) = +\infty$ pour chaque sommet s;
$d(v_{deb}) = 0$;
tant que *il y a un sommet hors de* P **faire**
  Choisir un sommet s hors de P de plus petite distance $d(s)$;
  Mettre s dans P;
  pour *chaque sommet* u *hors de* P *voisin de* s **faire**
    $d(u) = \min(d(u), d(s) + poids(s, u)$
  fin
fin

Exercice 14. On va montrer dans cet exercice que l'algorithme fait ce qu'on lui demande, à noter que dans cette version minimaliste on ne garde pas en tête le chemin le plus court mais juste la distance de ce chemin.

1. Montrer que l'algorithme termine.
2. Montrer par récurrence sur le cardinal de P qu'à chaque étape on a les deux propriétés suivantes
   (a) Pour tout $v \in P, d(v)$ est la distance du plus court chemin reliant v_{deb} à v.
   (b) Pour tout $v \notin P$, $d(v)$ est la distance du plus court chemin reliant v_{deb} à v ne passant que par des sommets de P à part v.

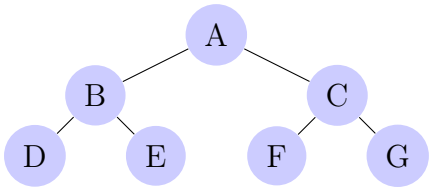
3. En déduire que l'algorithme est correct.
4. Montrer que la complexité de l'algorithme est en $O(|S|^2)$, on peut en fait descendre à $O((|S| + |A|) \log |S|)$.

Remarque 3.2. C'est ce genre d'algorithme qu'utilise Google Maps par exemple pour vous donnner le plus court chemin entre deux adresses.

3.2 Le parcours en profondeur

A la différence du parcours en largeur, ici on va aller de plus en plus profondément dans le graphe avant de revenir lorsque l'on est bloqué. Voici l'algorithme, celui-ci est récursif

```
Parcourir(graphe G, sommet v) =;
Marquer le sommet v;
Afficher v;
pour tout voisin t de s faire
    si t n'est pas marqué alors
        Parcourir(graphe G, sommet t)
    fin
fin
```



Exemple 3.3. Si on reprend l'arbre précédent le parcours en profondeur va donner A, B, D, E, C, F, G. Visuellement, on voit bien ce qu'il se passe, on parcourt chaque branche de l'arbre et une fois qu'on est arrivé au bout, on remonte et on redescend par une autre branche jusqu'à ce que l'on ait tout parcouru.

Exercice 15. Montrer que cet algorithme termine.

Exercice 16. Supposons donné G un graphe connexe et deux sommets u, v. Donner un algorithme utilisant un parcours en profondeur pour trouver un chemin entre u et v.

Exemple 3.4. Si on revient au problème du cavalier sur l'échiquier, une première étape peut déjà être de trouver un chemin entre deux cases. C'est ce que l'on fait avec ce genre d'algorithme.

Exercice 17. Donner un algorithme utilisant un parcours en profondeur qui, étant donné un graphe connexe, donne un chemin hamiltonien s'il en existe et retourne faux sinon.