Chapitre 8. Sommets d'un graphe

Un graphe est une modélisation d'un ensemble (non vide) d'objets reliés entre eux ; les objets sont appelés sommets, et les liens entre les objets sont appelés arêtes.

On peut utiliser les graphes pour représenter diverses situations courantes : les liens d'amitié entre étudiants, les matches entre équipes de sport, les liaisons moléculaires entre des atomes, les routes entre les villes, ...

Nous vous fournissons une définition python de la *classe* des graphes, ainsi que des autres classes (sommets et arêtes) nécessaires. Ces définitions se trouvent dans le module `bibgraphes.py` écrit par des enseignants, disponible en téléchargement sur le site https://moodle.u-bordeaux.fr/course/view.php?id=4641. Allez sur ce site, retrouvez dans la section du chapitre 8 le fichier `bibgraphes.py`, utiliser un clic droit dessus et utilisez "enregistrer sous" pour l'enregistrer à côté de vos autres fichiers python. Ce module comporte aussi une vingtaine de fonctions qui permettent de manipuler graphes, sommets et arêtes sans connaître les détails des classes correspondantes. Pour utiliser un module il faut commencer par l'*importer*, et toute session de travail sur les graphes doit commencer par la phrase magique :

```
from bibgraphes import *
```

La figure 8.1 montre un exemple de graphe : les sommets sont quelques grandes villes françaises, et une arête entre deux villes indique que des rames TGV circulent entre elles [a] Le détail du dessin, notamment la position géographique étrange de Nantes, est sans importance, la seule chose qui compte est que ce graphe comporte les 9 sommets et les 20 arêtes de la figure ci-contre.

a. Ce graphe correspond à la situation de l'année 2005 : Strasbourg est un sommet isolé, car la ligne TGV Est n'était pas encore en service à l'époque.

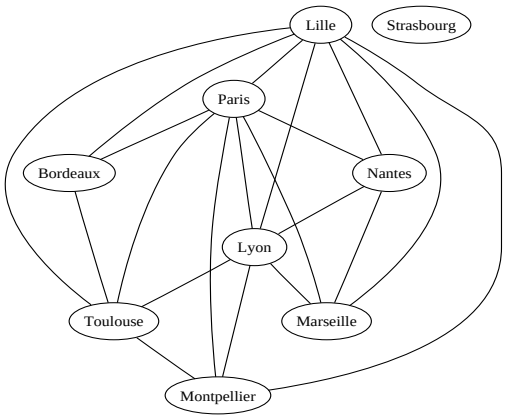

FIGURE 8.1 – Le graphe des lignes de TGV en 2005

On peut stocker toutes les informations d'un graphe dans un fichier de format `.dot`. Pour manipuler un graphe dans un programme python, on utilisera la fonction suivante :

`ouvrirGraphe(nom:str) -> graphe`	Ouvre le fichier *nom* et retourne le graphe contenu dedans.
`afficherGraphe(G:graphe)`	dessine le *graphe* `G`

Pour travailler sur le graphe du TGV (fichier `tgv2005.dot` disponible sur le site du cours), on commencera donc par l'instruction suivante :

```
tgv = ouvrirGraphe("tgv2005.dot")
```

Avec Python Tutor, le graphe apparait à l'écran. Si l'on utilise Spyder, il faut utiliser l'instruction suivante :

```
afficherGraphe(tgv)
```

Si dans une session python, à la suite de cette instruction, on tape simplement `tgv`, ou `print(tgv)`, l'interprète (en jargon : *Python Shell*) répond : `<graphe: 'tgv2005'>` pour indiquer que c'est un graphe dont le nom est `'tgv2005'`. Pour désigner le sommet *Bordeaux* on pourrait croire qu'il suffit de taper `Bordeaux`, hélas l'interprète répond, rouge de colère : `NameError: name 'Bordeaux' is not defined.`

En effet il n'y a pas de variable appelée `Bordeaux` qui contiendrait ce sommet. On peut obtenir la liste complète des sommets d'un graphe G avec la fonction `listeSommets(G:graphe)`. En reprenant le même exemple, si l'on tape `listeSommets(tgv)` on obtient :

```
[<sommet: 'Lille', 'white', False>, <sommet: 'Paris', 'white', False>, ...
 <sommet: 'Bordeaux', 'white', False>, ... <sommet: 'Strasbourg', 'white', False>]
```

Chaque sommet est affiché avec confirmation de sa classe, suivie du nom du sommet et de sa couleur — pour l'instant `"white"`, donc coloré en blanc ; la dernière composante est booléenne et indique si le sommet est marqué ou non : ces marques seront utilisées section 10.3, pour l'instant `False` est affiché partout car aucun sommet n'est marqué.

Le nom d'un sommet, par exemple `"Bordeaux"`, est une simple chaîne de caractères utilisée pour identifier ce sommet à l'intérieur du graphe lorsque l'on affiche le sommet ou lorsque l'on dessine le graphe. Comme on l'a expliqué ci-dessus, ce n'est pas le nom d'une variable, c'est juste une étiquette appliquée au sommet. Il faut utiliser la fonction `sommetNom(G:graphe,nom:str)` pour accéder au sommet par son nom : `sommetNom (tgv, "Bordeaux")` est une expression correcte pour désigner ce sommet, et on peut ensuite le stocker dans une variable en utilisant une affectation :

```
bx = sommetNom (tgv, "Bordeaux")
```

Pour bien distinguer les deux, nous avons choisi ici un nom de variable `bx` distinct de l'étiquette *Bordeaux*.

`listeSommets(G:graphe)->list`	retourne la *liste* des *sommets* de `G`
`nbSommets(G:graphe)->int`	retourne le *nombre* de sommets de `G`, c'est-à-dire la *taille* de la liste précédente
`sommetNom(G:graphe, etiquette:str)->sommet`	retourne le *sommet* de `G` désigné par son *nom* (`etiquette`), par exemple : `sommetNom (tgv, "Bordeaux")`
`nomSommet(s:sommet)->str`	retourne le *nom* du sommet `s`

Quelques graphes dont celui du TGV sont disponibles sur le site du cours https://moodle.u-bordeaux.fr/course/view.php?id=4641. Il est aussi possible de récupérer des graphes sur Internet. On pourra par exemple consulter la bibliothèque https://networkdata.ics.uci.edu/index.php qui contient d'autres exemples. Les formats supportés par le module `bibgraphes.py` sont .dot, .gml et .paj. Attention à la taille des graphes, certains contiennent de millions de sommets (*nodes*) ou d'arêtes (*edges*) et seront *très* longs à traiter !

8.1 Échauffement : coloriages

Un sommet s peut être colorié avec une couleur c par la fonction `colorierSommet(s:sommet, c:couleur)`, où c est une chaîne de caractères. La fonction `afficherGraphe(G:graphe)` tient compte des couleurs des sommets si celles-ci font partie d'une liste prédéfinie ; par exemple, `"red"`, `"green"`, `"blue"` sont des couleurs reconnues par le programme de dessin, et la liste complète se trouve à l'adresse : http://www.graphviz.org/doc/info/colors.html – vous y trouverez entre autres `"orange"`, `"chocolate"` et `"tomato"` !

`colorierSommet(s:sommet, c:str)`	colorie le *sommet* `s` avec la *couleur* `c` (par exemple `"red"`)
`couleurSommet(s:sommet)->str`	retourne la *couleur* du *sommet* `s`.
`afficherGraphe(G:graphe)`	dessine le *graphe* `G`

Dans les deux exercices suivants on suppose que la variable `tgv` contient le graphe du TGV (Figure 8.1).

Exercice 8.1.1 TP

Note : étudier le chapitre le 8 depuis le début pour avoir les explications sur le fonctionnement des graphes en python.

1. Stocker le sommet nommé *Paris* du graphe `tgv` dans une variable `p`.
2. Appeler la fonction `colorierSommet` en lui passant en paramètre la variable `p` et la couleur `"green"` (vert). Appeler la fonction `afficherGraphe` sur le graphe. Constater la coloration.
3. Colorier le sommet *Bordeaux* du graphe `tgv` en bleu, en une seule ligne, en combinant les appels de fonctions plutôt que passer par une variable. Relancer `afficherGraphe` pour constater la coloration.

Exercice 8.1.2 Puisque la fonction `listeSommets` retourne une liste des sommets du graphe, on peut l'utiliser au sein d'une boucle `for` pour effectuer une opération sur chacun des sommets du graphe.

Écrire une fonction `toutColorier(G:graphe,c:str)` qui colorie tous les sommets du graphe G avec la couleur c. Utiliser cette fonction pour colorier en rouge tous les sommets du graphe `tgv` ; vérifier le résultat de deux façons :

a) en affichant la liste des sommets du graphe,

b) en dessinant le graphe.

Écrire l'instruction qui permet d'annuler l'opération précédente, c'est-à-dire de recolorier les sommets en blanc ; vérifier que les sommets du graphe `tgv` sont bien decoloriés.

Exercice 8.1.3

1. Écrire une fonction `nbSommetsCouleur(G:graphe,c:str)->int` qui compte les sommets du graphe G qui ont la couleur c.
2. Écrire une fonction `nbSommetsColores(G:graphe)->int` qui compte les sommets colorés de G (c'est-à-dire les sommets qui ont une couleur différente de `"white"`) en *appelant* la fonction précédente et la fonction `nbSommets(G)`.

8.2 Voisins

Deux sommets s et t sont appelés voisins s'il existe une arête e ayant s et t comme extrémités ; on dit que l'arête e est incidente à chacun des sommets s et t.

Une boucle est une arête dont les deux extrémités sont confondues et correspondent au même sommet.

Par exemple, les sommets A et B du graphe ci-contre sont voisins, ainsi que A et C, tandis que les sommets A et D ne sont pas voisins. Il peut exister plusieurs arêtes entre deux sommets, par exemple ici entre A et B, on parle alors d'arête multiple. Sur le graphe ci-contre, il y a une boucle autour du sommet B et deux boucles autour du sommet D.

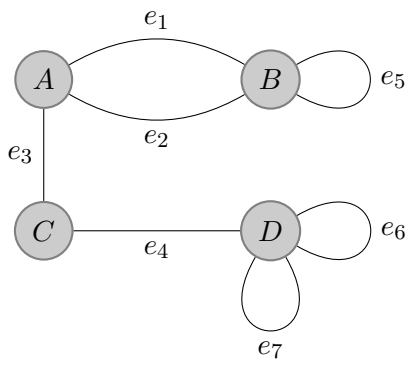

FIGURE 8.2 – un graphe avec une arête double et trois boucles

La fonction `listeVoisins(s:sommet)->list` retourne la liste des voisins du sommet s, obtenue en suivant chacune des arêtes incidentes. Un même sommet peut se retrouver plusieurs fois dans cette liste : par exemple dans le graphe figure 8.2, B apparaît deux fois dans la liste des voisins de A.

Une boucle autour du sommet s est, par convention, deux fois incidente à s : la liste des voisins de B contient deux fois le sommet B lui-même, à cause de la boucle autour de B (que l'on peut considérer dans un sens ou dans l'autre).

Le degré d'un sommet s est le nombre d'incidences d'arêtes, et la fonction `degre(s:sommet)` retourne sa valeur ; c'est aussi le nombre de brins issus de s lorsqu'on regarde le dessin — une boucle compte pour deux dans le calcul du degré. *Attention* : il ne faut jamais de lettre accentuée dans le nom d'une *fonction* python.

`listeVoisins(s:sommet)->list`	retourne la *liste* des *voisins* du sommet `s`
`degre(s:sommet)->int`	retourne le *degré* du sommet `s`, qui est aussi la *taille* de la liste des voisins

Exercice 8.2.1 Pour chaque sommet du graphe de la figure 8.2, noter sur papier son nom, son degré et écrire la liste de ses voisins (leur ordre dans la liste est sans importance). Écrire l'instruction python qui permet d'imprimer la même chose en TP (le fichier correspondant à ce graphe est `fig32.dot`).

Exercice 8.2.2 Appeler la fonction `listeVoisins` pour afficher la liste des villes voisines de *Nantes* dans le graphe du TGV.

Exercice 8.2.3 Écrire une fonction `colorierVoisins(s:sommet,c:str)` qui colorie tous les voisins du sommet s avec la couleur c. Utiliser cette fonction pour colorier en vert les voisins de *Bordeaux* dans le graphe du TGV. Vérifier le résultat de deux façons : afficher la liste des sommets du graphe, et l'afficher.

8.3 Calculs sur les degrés

On rappelle que le degré d'un sommet est le nombre d'arêtes incidentes à ce sommet. Une boucle compte pour deux dans le degré. Par exemple, dans le graphe de la figure 8.2 (reproduite ci-dessous) :

- le degré de A est 3, car A est extrémité de e_1, e_2 et e_3 ;

- le degré de B est 4, car B est extrémité de e_1, e_2 et e_5 (qui contribue pour 2 unités au degré, car c'est une boucle autour de B).

Exercice 8.3.1 Soit la fonction suivante :

```
def mystere(G):
  n = nbSommets(G)
  x = 0
  for s in listeSommets(G):
    x = x + degre(s)
  return x/n
```

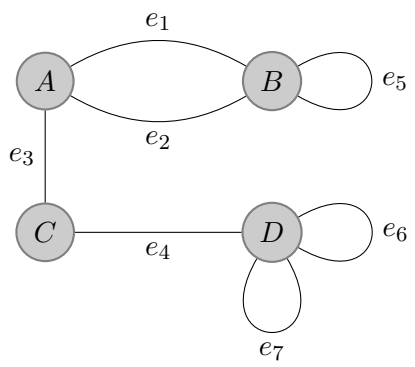

Figure 8.2 idem au graphe page 62.

1. Quelle est la valeur calculée si G est le graphe de la figure ci-contre ?
2. Que calcule la fonction `mystere` dans le cas général ?

Exercice 8.3.2 1. Écrire une fonction `degreMax(G:graphe)->int` qui calcule et retourne le maximum des degrés des sommets d'un graphe G.

2. Écrire une fonction `degreMin(G:graphe)->int` qui calcule et retourne le minimum des degrés des sommets d'un graphe G. Pourquoi cette question est-elle un peu plus difficile que la précédente ?

Note : On pourra utiliser `listeSommets(G)[0]` pour accéder au premier sommet du graphe.

Exercice 8.3.3 Écrire une fonction `nbSommetsDegre(G:graphe,d:int)->int` qui calcule et retourne le nombre de sommets du graphe G ayant pour degré d.

Exercice 8.3.4 TP Ouvrez le graphe *Power Grid* qui représente le réseau électrique américain, à télécharger depuis le site du cours (fichier `power.gml`). N'essayez pas de le faire afficher, il est très gros, cela prendrait beaucoup de temps !

En utilisant les fonctions de l'exercice précédents et la fonction `nbSommets` :

- Vérifier qu'il n'y a pas de sommet isolé (c'est-à-dire de degré 0).
- Compter le nombre de sommets de degré 1 et de sommets de degré 2.
- Calculer le degré maximum des sommets.
- Calculer la moyenne des degrés des sommets.

On peut ainsi avoir une idée de la robustesse du réseau électrique américain.

8.4 Exercices de révisions et compléments

Calculs sur les degrés

Exercice 8.4.1 TP Ouvrez le graphe *Les Miserables* qui représente la co-apparition des personnages du roman *Les Misérables*, à télécharger depuis le site du cours (fichier `lesmis.gml`).

Écrivez une fonction pour déterminer quel personnage interagit le plus avec d'autres personnages (la fonction `nomSommet(s:sommet)->str` retourne l'étiquette du sommet s). Il s'agira très probablement du personnage principal de l'œuvre. A-t-il pour autant croisé *Eponine* durant le récit ?

Exercice 8.4.2 TP Écrivez une fonction `moyenneDegresVoisins(s:sommet)->float` qui calcule la moyenne des degrés des voisins du sommet `s`. Écrivez une fonction `nbDegreMoindreVoisins(G:graphe)->int` qui compte le nombre de sommets pour lesquels leur degré est inférieur à la moyenne des degrés de leurs voisins. Appelez-la sur différents graphes et comparez à chaque fois au nombre de sommets. Écrivez une fonction `nbDegreMoindre(G:graphe)->int` qui compte le nombre de sommets dont le degré est inférieur à la moyenne des degrés des sommets du graphe (réutilisez la fonction de l'exercice 8.3.1). Que remarquez-vous ?

C'est un effet bien connu : sur les réseaux sociaux, on a l'impression d'avoir moins d'amis que nos amis ont d'amis, alors qu'en fait globalement non. C'est simplement parce que les personnes ayant beaucoup d'amis sont sur-représentées dans la moyenne ; c'est expliqué dans la vidéo Micmaths https://www.youtube.com/watch?v=MySkCFFgiRQ

Boucle conditionnelle

Exercice 8.4.3 TP Une marche aléatoire dans un graphe est un chemin construit aléatoirement : partant d'un sommet initial, on tire au hasard le sommet suivant parmi ses voisins et, en répétant $(k-1)$ autres fois ce processus, on obtient un chemin de longueur k.

Pour choisir au hasard un sommet parmi la liste des voisins d'un sommet `s` vous pourrez utiliser l'instruction : `elementAleatoireListe(listeVoisin(s))`

Dans cet exercice, on cherche à mesurer expérimentalement, pour un graphe et un sommet donnés, la longueur moyenne des marches aléatoires qui reviennent au sommet de départ.

1. Écrire une fonction `longueurCycleAleatoire(G:graphe,s:sommet)->int` qui retourne la longueur d'une marche aléatoire partant du sommet `s` et s'arrêtant dès que l'on revient sur ce même sommet.

2. Tester votre fonction sur un chemin `"s0"` – `"s1"` – ... – `"s9"` de longueur 10 créé à l'aide de l'appel `construireGrille(1,10)`. On pourra afficher le nom des sommets visités successivement.

3. Écrire une fonction `moyenneCycleAleatoire(G:graphe,etiquette:str,k:int)->int` qui retourne la longueur moyenne de `k` marches aléatoires formant un cycle et partant du sommet initial dont le nom est `etiquette`.

4. Tester ainsi votre programme sur un chemin de longueur 10 :

   a) `moyenneCycleAleatoire(construireGrille(1,10), "s0", 1000)`
   b) `moyenneCycleAleatoire(construireGrille(1,10), "s2", 1000)`
   c) `moyenneCycleAleatoire(construireGrille(1,10), "s4", 1000)`

   Commenter et expliquer les résultats obtenus.

8.5 L'essentiel du chapitre

Un graphe est une modélisation d'un ensemble (non vide) d'objets reliés entre eux ; les objets sont appelés sommets, et les liens entre les objets sont appelés arêtes.

On peut ouvrir un graphe et l'afficher :

```
tgv = ouvrirGraphe("tgv2005.dot")
afficherGraphe(tgv)
```

Il y a une différence entre un sommet et som nom, il existe deux fonctions pour passer de l'un à l'autre :

```
bdx = sommetNom(tgv,"Bordeaux")
n = nomSommet(bdx)
```

On peut colorier les sommets et récupérer leur couleur (écrite sous forme d'une chaîne de caractères) :

```
colorierSommet(bdx, "red")
c = couleurSommet(bdx)
```

On peut parcourir tous les sommets d'un graphe :

```
for s in listeSommets(tgv):
    colorierSommet(s, "red")
```

Les sommets ont un degré qui correspont à l'incidence des arêtes qui sont autour :

```
print(degre(bdx))
```

On peut parcourir les voisins d'un sommet :

```
for s in listeVoisins(bdx):
    colorierSommet(s, "blue")
```