# Chapitre 10. Formule des poignées de mains, graphes simples

## 10.1 Formalisation pour les graphes

En informatique, on est souvent amené à faire des démonstrations de propriétés. L'objectif est d'établir sa véracité (ou sa fausseté). Lorsque celle-ci est quantifiée de manière existentielle ($\exists$) il suffirait d'exhiber le cas favorable, mais ce n'est pas toujours aisé. Lorsqu'elle est quantifiée de manière universelle ($\forall$), et que l'ensemble de référence est grand, il faut mettre en œuvre une démonstration puisqu'énumérer toutes les situations serait contre-productif.

On peut distinguer deux grands types de preuves :

1. La preuve directe : ce que l'on doit prouver est une conséquence de propriétés connues. On peut progresser dans le raisonnement petit à petit, en utilisant les propriétés et définitions connues. Pour progresser on peut également utiliser un thèorème : on vérifie que ses conditions sont bien remplies, on peut alors utiliser son résultat pour continuer.

2. La preuve par l'absurde : pour établir une propriété, on fait l'**hypothèse** que la négation de la propriété à démontrer est vraie. On effectue à partir de cette hypothèse un raisonnement qui conduit à une contradiction, on dit aussi incohérence, ou absurdité. Cela conduit donc à invalider l'hypothèse qui avait été faite, et donc que sa négation est vraie c'est-à-dire que la propriété qui nous intéressait est vraie.

Pour pouvoir produire des démonstrations sur les graphes, nous posons ici un formalisme.

On note $S(G)$ l'ensemble des sommets du graphe $G$ et $A(G)$ l'ensemble des arêtes du graphe $G$.

Ainsi on pourra écrire des énoncés sur les graphes, par exemple :

$$\forall s \in S(G), \text{degré}(s) \geq 1$$

Par ailleurs, le cardinal d'un ensemble $E$ est le nombre d'éléments que contient cet ensemble. On le note couramment "$|E|$". Ainsi, $|S(G)|$ dénote le cardinal de l'ensemble des sommets du graphe $G$, c'est-à-dire le nombre de sommets de $G$, et $|A(G)|$ est le nombre d'arêtes de $G$.

## 10.2 Formule des poignées de mains

**Exercice 10.2.1**

1. Dessiner un graphe à 6 sommets tel que la liste des degrés des sommets soit :
   `[2, 1, 4, 0, 2, 1]`.

2. Même question avec `[2, 1, 3, 0, 2, 1]`.

**Exercice 10.2.2** Un graphe est dit cubique si tous ses sommets sont de degré 3.

1. Dessiner un graphe cubique ayant 4 sommets ; même question avec 3 sommets, 5 sommets. Que constate-t-on ?

2. Quel est le nombre d'arêtes d'un graphe cubique de $n$ sommets ?

3. En déduire qu'un graphe cubique a un nombre pair de sommets.

On peut généraliser ce raisonnement sous la forme d'une relation liant le nombre d'arêtes d'un graphe à la somme des degrés de ses sommets. Cette relation, appelée *formule générale des poignées de mains*, fait partie des résultats à mémoriser :

$$\sum_{s \in S(G)} \text{degré}(s) = 2|A(G)|$$

**Exercice 10.2.3** TP

En utilisant la formule de poignées de mains, écrire une fonction `nbAretes(G:graphe)->int` qui calcule et retourne le nombre d'arêtes d'un graphe $G$.

Appliquer votre fonction `nbAretes` au graphe `fig32.dot` de la figure 8.2 page 62. Vérifier que votre fonction calcule correctement le nombre d'arêtes.

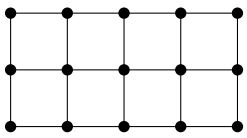
### 10.2.1 Exercices de révisions et compléments

**Exercice 10.2.4 (extrait DS 2016-2017)**

1. Un graphe peut-il avoir un seul sommet de degré impair ? Justifier.

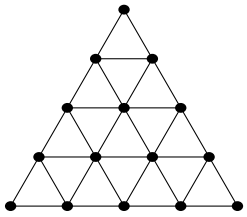
2. Quelle propriété mathématique (simple) possède le nombre de sommets de degré impair d'un graphe ? Justifier.

**Exercice 10.2.5** Une grille (rectangulaire) $m \times n$ est constituée de $m$ lignes horizontales et $n$ lignes verticales, dont les croisements forment les sommets du graphe ; une grille $3 \times 5$ est représentée ci-contre.



1. Compter les sommets de degré 2 (respectivement 3 et 4) et en déduire la somme des degrés des sommets en fonction de $m$ et $n$.

2. Compter les arêtes et comparer avec le résultat de la question précédente.



**Exercice 10.2.6** La grille triangulaire $T_5$ est représentée ci-contre. Adapter les calculs de l'exercice précédent au cas de $T_n$.

**Exercice 10.2.7** On dispose de 235 machines, que l'on souhaite relier en réseau. Est-il possible de relier chaque machine à exactement 17 autres machines ? Justifier la réponse, soit en expliquant comment relier les machines, soit en expliquant pourquoi ce n'est pas possible.

## 10.3 Graphes simples

Dans cette section, un graphe est dit simple s'il n'a ni boucle, ni arête multiple ; autrement dit, les extrémités d'une arête sont toujours distinctes, et il existe au plus une arête qui relie deux sommets $s$ et $t$ distincts. Par exemple, le graphe du TGV (Figure 8.1) est simple, et celui de la figure 8.2 ne l'est pas ; les graphes simples sont de loin les plus fréquents en pratique.

**Exercice 10.3.1**

1. Essayer de construire un graphe simple ayant 4 sommets, et tel que les degrés des sommets soient tous distincts.

2. On se propose de démontrer par l'absurde qu'il n'existe pas de graphe simple de $n$ sommets ($n \geq 2$) tel que tous ses sommets soient de degrés distincts. Supposons qu'un tel graphe $G$ existe.

   a) Montrer que $G$ ne peut comporter de sommet de degré supérieur ou égal à $n$.

   b) En déduire les degrés possibles pour les $n$ sommets.

   c) Montrer que ceci entraîne une absurdité.

3. Application : en déduire que dans un groupe de $n$ personnes, il y en a toujours deux qui ont le même nombre d'amis présents.

Dans un graphe simple la liste des voisins d'un sommet $s$ ne comporte jamais de répétition, c'est même une propriété caractéristique des graphes simples.

**Exercice 10.3.2** Écrire une fonction `nbOccurrences(L:list, x)->int` qui renvoie le nombre d'occurrences de l'élément `x` dans la liste `L`, c'est-à-dire le nombre de fois où il apparaît dans la liste.

Écrire une fonction `estSimple(G:graphe)->bool` qui teste si le graphe $G$ est simple, en s'aidant de la fonction `nbOccurrences` pour vérifier qu'un sommet n'apparaît pas deux fois dans une liste de voisins.

**Exercice 10.3.3** TP Expérimentez dans Python Tutor :

```
L1 = [1,2,3]
L2 = [5,6,7]
L = L1 + L2
LL = []
LL = LL + [0]
LL = LL + [1]
```

L'opérateur + sait donc "ajouter" des listes : il fabrique une nouvelle liste qui contient d'abord les éléments de la liste de gauche, puis ceux de la liste de droite. On peut ainsi par exemple construire la liste des entiers de 1 à n :

```
L=[]
for i in range(1,n+1):
  L = L + [i]
```

**Exercice 10.3.4** On raffine ici l'exercice 10.3.2 pour le cas des graphes non simples. On souhaite désormais déterminer quels sommets sont adjacents à des boucles ou des arêtes multiples.

Écrire une fonction `listeSommetsVoisinsRepetes(G:graphe)->list` qui retourne la liste des sommets du graphe $G$ pour lesquels un sommet apparaît au moins deux fois dans les listes des voisins de ces sommets.
Commencer par écrire une fonction `existeVoisinsRepetes(s:sommet) -> bool` qui teste si le sommet `s` a au moins deux fois le même sommet dans la liste de ses voisins. Utiliser cette fonction pour écrire `listeSommetsVoisinsRepetes`.

### 10.3.1 Exercices de révisions et compléments

**Exercice 10.3.5** Un graphe *complet* est un graphe où tout sommet est relié à chacun des autres par une arête. Évaluer le nombre de comparaisons nécessaires à la fonction `estSimple(G)` lorsque `G` désigne un graphe complet à $n$ sommets.

Une méthode plus efficace pour tester si tous les sommets d'une liste sont distincts est de marquer chaque sommet en parcourant la liste, et si l'on rencontre un sommet déjà marqué pendant ce parcours on sait que ce sommet est présent plusieurs fois dans la liste.

Cette méthode comporte un piège, car un sommet marqué le reste ! Si un utilisateur applique deux fois la fonction à la même liste on va trouver, lors de la seconde passe, que le premier sommet est marqué, et on va en déduire bêtement qu'il s'agit d'un sommet répété. Il faut donc commencer par démarquer tous les sommets avant d'appliquer l'algorithme. Les fonctions disponibles pour manipuler les marques sur les sommets sont :

| | |
|---|---|
| `marquerSommet(s:sommet)`<br>`demarquerSommet(s:sommet)` | marque le sommet `s`<br>démarque le sommet `s` |
| `estMarqueSommet(s:sommet)->bool` | retourne `True` si `s` est marqué, `False` sinon |

Note : le marquage d'un sommet est indépendant de la coloration d'un sommet.

**Exercice 10.3.6**

1. Marquez le sommet `Bordeaux` du graphe du TGV. Dessinez le graphe pour observer comment c'est illustré.

2. Écrire une fonction `demarquerVoisins(s:sommet)` qui démarque tous les voisins du sommet $s$.

3. Écrire une fonction `voisinsDistincts(s:sommet)->bool` qui teste si tous les voisins du sommet $s$ sont distincts en utilisant l'algorithme expliqué plus haut — le résultat est `True` ou `False`. Tester la fonction sur chaque sommet $s$ du graphe de la figure 8.2, et afficher la liste des voisins de $s$ après chaque test pour repérer les sommets marqués (`True` ou `False` apparait à droite) et vérifier que ce sont bien les sommets prévus.

4. Utiliser les fonctions précédentes pour écrire une nouvelle version de la fonction `estSimple(G:graphe)->bool` qui teste si un graphe $G$ est simple.

Tester la fonction `estSimple` sur quelques-uns des graphes disponibles sur le site du cours. Après exécution du test sur un graphe $G$, afficher $G$ et/ou afficher la liste de ses sommets pour repérer ceux qui sont marqués ; interpréter le résultat, en particulier pour les graphes simples.

## 10.4 Exercices de révision et compléments

**Exercice 10.4.1** Écrire une fonction `listeVoisinsCommuns (s1:sommet,s2:sommet)->list` qui calcule et retourne la liste des voisins communs à deux sommets $s1$ et $s2$.

En utilisant un marquage, écrivez-en une version qui a une bonne complexité.

Écrire une fonction `trajet1Correspondance (s1:sommet,s2:sommet)->bool` qui renvoie `True` ou `False` selon qu'il était possible en 2005 d'aller en TGV d'une ville de sommet `s1` à une ville de sommet `s2` en effectuant au plus une correspondance.

## 10.5 L'essentiel du chapitre

On note $S(G)$ l'ensemble des sommets du graphe et $A(G)$ l'ensemble des arêtes du graphe. Le cardinal d'un ensemble est noté avec "|...|".

La formule générale des poignées de mains relie les degrés des sommets au nombre d'arêtes du graphe :

$$\sum_{s \in S(G)} \text{degré}(s) = 2|A(G)|$$

On peut prouver des énoncés

- soit directement, en utilisant des propriétés et définitions connues et en appliquant éventuellement des théorèmes.
- soit par l'absurde, en supposant que la négation de ce que l'on souhaite démontrer est vrai, et en montrant alors que cela nous conduit à une contradiction.

On peut "ajouter" des listes avec l'opérateur +. On peut ainsi construire des listes progressivement, par exemple :

```
L=[]
for i in range(1,n+1):
  L = L + [i]
```