Chapitre 13. Graphes eulériens (hors-programme)

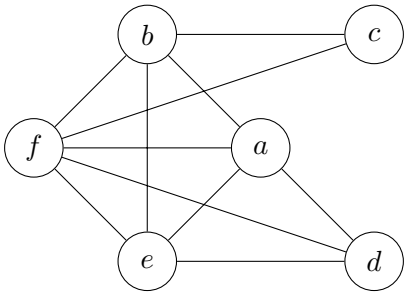
13.1 Définitions

Une chaîne est eulérienne si elle passe par chaque arête une fois et une seule et parcourt tous les sommets. Un graphe est dit eulérien s'il possède une chaîne eulérienne (ce peut être un cycle eulérien, c'est-à-dire une chaîne dont les extrémités sont confondues).

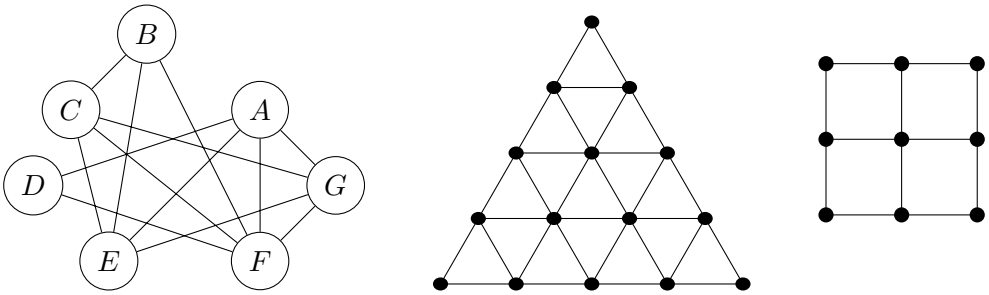
Exercice 13.1.1 Les propositions suivantes sont-elles vraies ou fausses ?

1. Tous les graphes eulériens sont connexes.
2. Si un graphe n'est pas eulérien, alors il n'est pas connexe.
3. Il existe des graphes connexes non eulériens.

Exercice 13.1.2 Le graphe ci-dessous admet-il une chaîne eulérienne ? Si oui, donner une telle chaîne, sinon, justifier. Admet-il un cycle eulérien ?



Exercice 13.1.3 Parmi les 3 graphes suivants, lesquels sont eulériens ?



Théorème 13.1.1 *Si un graphe admet un cycle eulérien, tous ses sommets sont de degré pair ; s'il admet une chaîne eulérienne d'extrémités distinctes, il a exactement 2 sommets de degré impair, qui sont les extrémités de la chaîne eulérienne.*

Exercice 13.1.4 L'affirmation suivante :

> *« si un graphe G admet un cycle eulérien, alors toutes les chaînes eulériennes de G sont des cycles »*

est-elle correcte ? Justifier ou donner un contre-exemple.

On admettra la réciproque du théorème 13.1.1 :

Théorème 13.1.2 *Si un graphe connexe a 0 ou 2 sommets de degré impair, il est eulérien.*

Exercice 13.1.5 Un graphe complet est un graphe où tout sommet est relié à chacun des autres par une arête. Pour quelles valeurs de n un graphe complet de n sommets (noté K_n) est-il eulérien ?

Exercice 13.1.6 On considère un graphe G eulérien dont les sommets sont A, B, C, D, E, F. On connaît une chaîne eulérienne de ce graphe :

$$[B, e_3, F, e_0, A, e_1, E, e_2, F, e_4, D, e_6, C, e_7, B, e_5, D]$$

1. Déduire de la chaîne eulérienne le degré de chaque sommet.
2. Dessiner G sans oublier de porter les noms des sommets et des arêtes, et en évitant les croisements d'arêtes (c'est possible : on dit que le graphe G est *planaire*).

Exercice 13.1.7 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 13.1.

Ainsi, un visiteur qui se trouve dans la salle S peut atteindre directement les salles A, D ou G. S'il se trouve dans la salle C, il peut se rendre directement dans la salle B, mais pas dans les autres salles. On s'intéresse au parcours d'un visiteur dans ce musée. On ne se préoccupe pas de la manière dont le visiteur accède au musée ni comment il en sort. Cette situation peut être modélisée par un graphe, les sommets étant les noms des salles, les arêtes représentant les portes de communication.

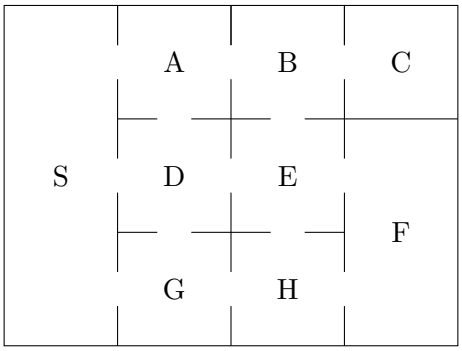


FIGURE 13.1 – Plan du musée

1. Dessiner un graphe modélisant la situation décrite.
2. Est-il possible de visiter le musée, en empruntant chaque porte une fois et une seule ? (justifier), sinon, comment le rendre possible ? Où placer l'entrée et la sortie ?

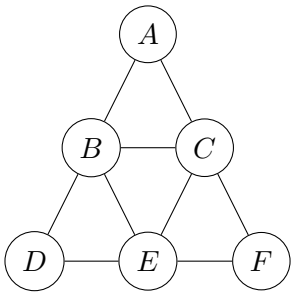
Dans la suite de ce chapitre, on se propose de démontrer une partie du théorème 13.1.2 : si tous les sommets d'un graphe connexe G sont de degré pair, alors G admet un cycle eulérien.

13.2 Construction d'un cycle eulérien

Soit un graphe connexe G dont tous les sommets sont de degré pair, voici un algorithme qui construit un cycle eulérien Γ :

1. choisir un sommet s arbitraire ;
2. former la chaîne $\Gamma = \{s\}$;
3. tant que le dernier sommet de Γ possède une arête incidente a qui n'appartient pas à Γ, ajouter à la chaîne Γ l'arête a et son sommet extrémité ;
4. si toutes les arêtes de G sont dans Γ alors l'algorithme est terminé et Γ est un *cycle eulérien* ;
5. sinon considérer le graphe G' constitué des arêtes qui ne sont pas dans Γ (et de leurs extrémités) ; choisir un sommet s' qui soit à la fois dans Γ et dans G' et construire dans G' un cycle Γ' d'origine s' avec le même algorithme ; enfin insérer Γ' dans Γ et retourner à l'étape précédente.

Exercice 13.2.1 Appliquer l'algorithme au graphe ci-contre, en choisissant toujours à l'étape 2 l'arête a dont l'extrémité porte le nom le plus petit dans l'ordre alphabétique, et en choisissant le sommet A comme sommet de départ à l'étape 1. Recommencer en faisant varier le sommet de départ.



Exercice 13.2.2 Soit G un graphe quelconque auquel l'algorithme soit applicable — G est donc connexe et tous ses sommets sont de degré pair.

1. Montrer qu'à la fin de l'étape 2 la chaîne Γ est forcément un cycle.
2. Montrer qu'à l'étape 4 tous les sommets du graphe G' sont de degré pair, et qu'il existe toujours un sommet s' qui appartient à la fois à Γ et à G'.
3. Construire un exemple tel que le graphe G' obtenu à l'étape 4 ne soit pas connexe.

Le module de manipulation de graphes contient les fonctions suivantes :

`listeAretesIncidentes(s)`	retourne la *liste* des arêtes *incidentes* à `s`
`sommetVoisin(s,a)`	retourne le voisin de `s` en suivant l'arête `a`, c'est-à-dire le sommet t extrémité de l'arête a (qui relie s et t)
`marquerArete(a)` `demarquerArete(a)`	marque l'arête `a` démarque l'arête `a`
`estMarqueeArete(a)`	retourne `True` si l'arête `a` est marquée, `False` sinon

Pour récupérer un sommet arbitraire d'un graphe G, vous pouvez utiliser la fonction `elementAleatoireListe(L:list)` (avec `L` une liste d'éléments) appliquée à la liste des sommets du graphe G, *cf.* aide-mémoire page 139 en fin de fascicule.

Exercice 13.2.3

1. Écrire une fonction `toutDemarquer(G:graphe)` qui démarque toutes les *arêtes* de G.
2. Écrire une fonction `areteIncidenteNonMarquee(s:sommet)` qui retourne une arête non marquée d'extrémité s s'il en existe une — sinon la fonction retourne `None`.
3. Écrire une fonction `cycleSansIssue(s:sommet)` qui exécute les *étapes 2 et 3* de l'algorithme et retourne le cycle d'origine s ainsi construit. Pour cela marquer les arêtes au fur et à mesure qu'elles sont ajoutées à la chaîne Γ (représentée par une liste).

Tester cette fonction sur le premier sommet de la grille triangulaire T_5 — voir exercices 10.2.5 et 13.1.3 (graphe du milieu) — que l'on construira avec la fonction `construireTriangle(n:int)`. Avec de la chance le cycle calculé est eulérien ; dans d'autres cas il manque des arêtes.

4. Combien d'arêtes le graphe T_5 possède-t-il ?
5. En déduire la taille d'un cycle eulérien de T_5 (ne pas oublier qu'un cycle comporte alternativement des sommets et des arêtes).
6. Pour chaque choix du sommet de départ s afficher la taille de la liste u retournée par la fonction `cycleSansIssue(s:sommet)` — la fonction `len(u)` fournit la taille d'une liste u ; lorsqu'il manque des arêtes afficher le graphe :
   a) les arêtes marquées apparaissent avec une épaisseur et une couleur différentes des arêtes non marquées : identifier le graphe G' (étape 4 de l'algorithme) et un sommet s' défini comme dans l'algorithme ;
   b) calculer `u2 = cycleSansIssue(s2)` *sans effacer le marquage* des arêtes de u — s2 désigne le sommet s' de la question précédente (s' n'est pas un nom valide pour python) ;
   c) recommencer tant qu'il existe des arêtes non marquées !

Exercice 13.2.4

1. Utiliser la fonction `melange` comme dans l'exercice 11.6.1 pour modifier la fonction `areteIncidenteNonMarquee` afin de retourner une arête choisie arbitrairement parmi les arêtes incidentes non marquées.
2. Choisir un sommet arbitraire s du graphe T_5 (pour cela vous pouvez, si vous le souhaitez, tirer avantage de la fonction `elementAleatoireListe`, *cf.* aide-mémoire page 139) et calculer plusieurs fois `cycleSansIssue(s)` ; après chaque calcul afficher le graphe et identifier le graphe G' (étape 4 de l'algorithme).

   Note : ne pas oublier d'utiliser la fonction `toutDemarquer` avant de lancer un nouveau calcul.

Exercice 13.2.5 Écrire une fonction `sommetAvecAretesIncidentesMarqueeEtNonMarquee(G:graphe)` qui renvoie un sommet de `G` ayant à la fois une arête incidente marquée et une arête incidente non marquée. Si un tel sommet n'existe pas, la fonction renverra `None`.

Exercice 13.2.6 Écrire une fonction `listeCyclesSansIssue(G:graphe)` qui renvoie une liste de cycles sans issue composant un cycle eulérien (dans le cas ou le graphe `G` possède un tel cycle). Si `G` ne possède pas de cycle eulérien, la fonction renverra la liste vide.

Exercice 13.2.7 Écrire une fonction `cycleEulerien(G:graphe)` qui renvoie (sous forme de liste) un cycle eulérien dans `G`. Si le graphe ne contient pas de cycle eulérien, la fonction renverra la liste vide. Pour écrire cette fonction, vous pourriez tirer profit de fonctions préexistantes travaillant sur des listes d'éléments, *cf.*, https://docs.python.org/3/tutorial/datastructures.html

Exercice 13.2.8 Soit `G` un graphe. Écrire une fonction `calculeCycleEulerienOuChaineEulerienne(G:graphe)`. Si le graphe n'est pas eulérien, la fonction renverra la valeur `None`. Si le graphe est eulérien, la fonction renverra une chaîne eulérienne ou un cycle eulérien sous forme d'une liste ; en outre, la fonction marquera et coloriera les arêtes du cycle eulérien ou de la chaîne eulérienne. Vous pourrez écrire deux versions de cette fonction. La première version n'utilisera pas les fonctionnalités de l'exercice 13.2.4, tandis que le seconde version `calculeCycleEulerienOuChaineEulerienneV2(G:graphe)` les utilisera. Tester plusieurs fois chacune de vos fonctions sur différents graphes.