Informatique

TP X4

Compléments sur les graphes simples : Forêts couvrantes

1 Introduction

Soit $G = (V, E)$ un graphe simple.

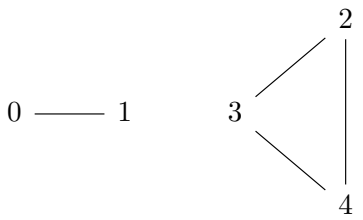
1.1 Chaînes simples

On dit qu'une arête $\{u, v\} \in E$ **apparaît** dans une chaîne $v_0, \ldots, v_k$ s'il existe (au moins) un $i < k$ tel que ($u = v_i$ et $v = v_{i+1}$) ou ($v = v_i$ et $u = v_{i+1}$).

Exemple 1.1. *Considérons les chaînes*

$$0, 1 \qquad 0, 1, 0 \qquad 2, 3, 4 \qquad \textit{et} \qquad 2, 3, 4, 2$$

dans le graphe suivant :



L'arête $\{0, 1\}$ *apparaît une fois dans la chaîne* $0, 1$ *et deux fois dans la chaîne* $0, 1, 0$ *(une fois dans le sens* $0 \to 1$*, et une fois dans le sens* $1 \to 0$*).*

L'arête $\{2, 4\}$ *n'apparaît pas dans la chaîne* $2, 3, 4$*, mais elle apparaît dans la chaîne* $2, 3, 4, 2$ *(dans le sens* $4 \to 2$*).*

Une chaîne $v_0, \ldots, v_k$ est **simple** si toute arête apparaît au plus une fois dans $v_0, \ldots, v_k$. Autrement dit, une chaîne $v_0, \ldots, v_k$ est simple si toutes ses arêtes sont distinctes (si $i \neq j$, alors $\{v_i, v_{i+1}\} \neq \{v_j, v_{j+1}\}$).

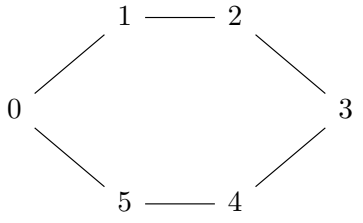
Question 1. *Parmi les chaînes de l'Exemple 1.1, lesquelles sont simples ?* Q.1

1.2 Cycles

Un **cycle** est une chaîne simple $v_0, \ldots, v_k$ telle que $k \geq 1$ et $v_0 = v_k$. Autrement dit, un cycle est une chaîne simple de longueur ≥ 1 entre un sommet et lui même.

Un graphe est **acyclique** s'il ne contient pas de cycle, c'est-à-dire si aucune de ses chaînes simples n'est un cycle.

Exemple 1.2. *La chaîne* $0, 1, 2, 3, 4, 5, 0$ *est un cycle dans le graphe*



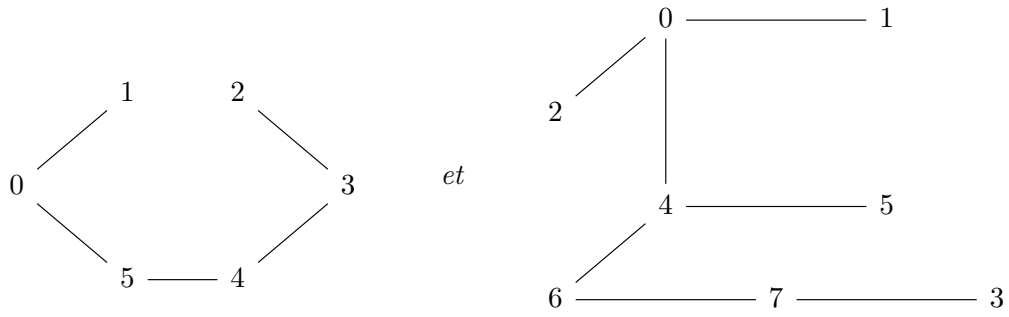
Question 2. *Parmi les chaînes de l'Exemple 1.1, lesquelles sont des cycles ?* Q.2

Remarque 1.3. *Un cycle doit comporter au moins trois sommets distincts.*

1.3 Arbres et forêts

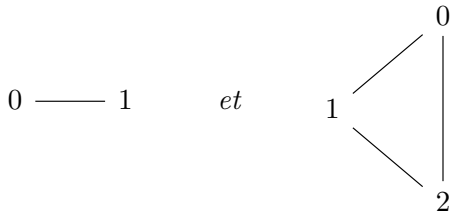
Une **forêt** est un graphe (simple) acyclique. Un **arbre** est un graphe (simple) connexe et acyclique. Ainsi, chaque composante connexe d'une forêt est un arbre.

Exemple 1.4. *Les deux graphes suivants sont des arbres :*



Question 3. *Donner des listes d'adjacences pour chacun des deux arbres de l'Exemple 1.4.* Q.3

Question 4. *Indiquer lesquels des graphes suivants sont des arbres :* Q.4

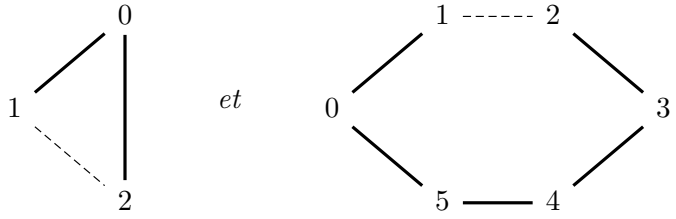


2 Forêts couvrantes

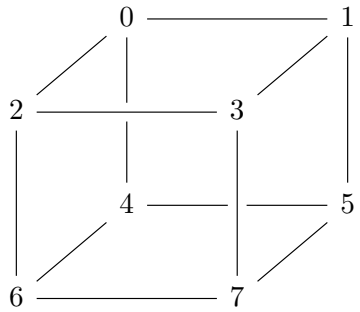
2.1 Arbres couvrants de graphes connexes

Soit $G = (V, E)$ un graphe (simple) connexe. Un **arbre couvrant** de G est un arbre $A = (V, T)$ avec $T \subseteq E$. Autrement dit, un arbre couvrant de G est un arbre qui a exactement les mêmes sommets que G et dont toutes les arêtes sont des arêtes de G.

Exemple 2.1. *Voici deux arbres couvrants, chacuns pour un graphe vu précédemment (les arêtes des arbres couvrants sont en* ***gras****, les autres arêtes sont en pointillés) :*



Question 5. *Donner un arbre couvrant pour le cube* Q.5

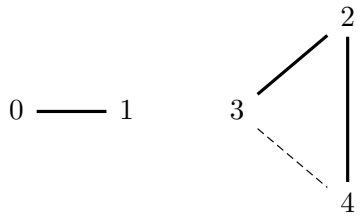


2.2 Forêts couvrantes de graphes non nécessairement connexes

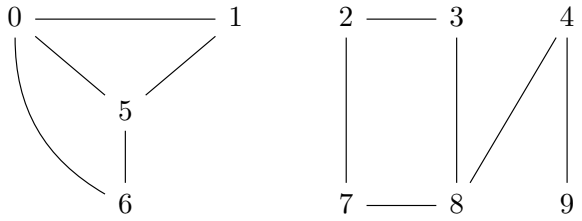
Soit $G = (V, E)$ un graphe (simple). Une **forête couvrante** de G est une forêt $F = (V, T)$ telle qu'un ensemble de sommets $A \subseteq V$ est une composante connexe de F si, et seulement si, $(A, T \cap A)$ est un arbre couvrant d'une composante connexe de G.

Ainsi, les sommets de F sont exactement les sommets de G, et toute arête de F est une arête de G. De plus, chaque arbre de F couvre une composante connexe de G, et chaque composante connexe de G est couverte par un arbre de F.

Exemple 2.2. *Voici une forêt couvrante du graphe de l'Exemple 1.1 :*



Question 6. *Donner une forêt couvrante du graphe* Q.6



2.3 Construction d'arbres couvrants

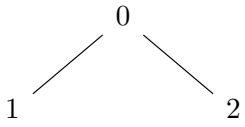
Nous allons maintenant utiliser les parcours de graphes pour obtenir des forêts couvrantes.

L'idée est la suivante. Soit $G = (V, E)$ un graphe simple, avec $V = \{0, \ldots, n-1\}$, et soit u un sommet de G. On cherche à construire un arbre couvrant de la composante connexe de u dans G. Pour ce faire, on se donne une liste `parent` de longeur n. Cette liste `parent` est initialisée à `None`. On parcourt G depuis le sommet u, et lors de ce parcours, à chaque fois qu'on explore une liste de voisins `lst`$[a]$, pour chaque sommet b (non encore visité) de cette liste, on met `parent`$[b]$ à a. Autrement dit, lorsqu'on traverse pour la première fois une arête $\{a, b\} \in E$ dans le sens $a \to b$, on indique dans `parent`$[b]$ que le sommet b a été découvert en venant du sommet a.

En fin d'algorithme, on obtient une liste `parent` qui contient uniquement `None` et des entiers parmi $0, \ldots, n-1$. Si `parent`$[i] \neq i$ pour tout $i = 0, \ldots, n-1$, alors une telle liste `parent` définit un graphe P de sommets $\{0, \ldots, n-1\}$ et tel que

$$\{a, b\} \text{ est une arête de } P \qquad \text{si et seulement si} \qquad (\texttt{parent}[a] = b \ \text{ ou } \ \texttt{parent}[b] = a)$$

Exemple 2.3. *Pour* `parent` $= [\texttt{None}, 0, 0]$*, le graphe* P *est*



Question 7. *Donner le graphe* P *pour* `parent` $= [\texttt{None}, 0, 0, 7, 0, 4, 4, 6]$. Q.7

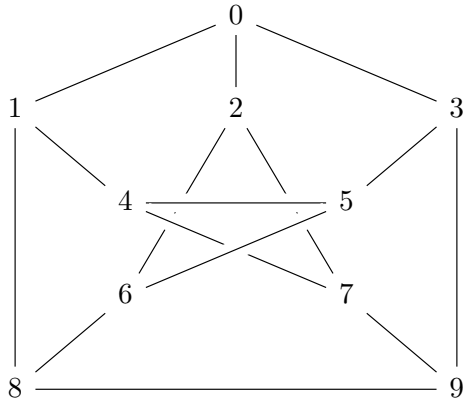
Question 8. *Écrire une fonction* Python *qui prend arguement une liste* `parent` *comme ci-dessus, et qui renvoie une liste d'adjacence pour le graphe* P *correspondant.* Q.8

On attend une complexité linéaire en la longueur de `parent`.

Question* 9. *Écrire une fonction* Python *qui prend en arguments un graphe* G *est qui renvoie une forêt couvrante de* G*.* **On pourra utiliser une liste `parent` comme ci-dessus.** *Q.9

On attend une complexité linéaire en la taille de G**.**

Question 10. *Utiliser la fonction de la Question 9 pour donner un arbre couvrant du graphe de Petersen :* Q.10



2.4 Détection de cycles

Nous allons maintenant voir que l'algorithme de la Question 9 donne une méthode très simple pour détecter si un graphe (simple) est acyclique. On ne considère que des graphes connexes. Rappelons qu'un tel graphe est acylique si, et seulement si, c'est un arbre.

Soit donc $G = (V, E)$ un graphe simple connexe. et considérons la forêt P renvoyée par l'algorithme de la Question 9. Comme G est connexe, P est en fait un arbre couvrant de G. De plus, on peut voir que P a exactement $|V| - 1$ arêtes.

Comme toutes les arêtes de P sont des arêtes de G, si $|E| = |V| - 1$ alors $G = P$, et G est acyclique. Supposons maintenant que G est acyclique, et considérons une arête $\{u, v\} \in E$ de G. Si cette arête n'est pas dans P, alors la rajouter à P crée un cycle (en effet, il existe une chaîne simple entre u et v dans P, et cette chaîne ne contient pas l'arête $\{u, v\}$ par hypothèse). Ainsi, toute arête de G est une arête de P, et $|E| = |V| - 1$.

Nous avons donc le résultat suivant.

Théorème 2.4. *Un graphe (simple) connexe* $G = (V, E)$ *est acyclique si, et seulement si,* $|E| = |V| - 1$*.*

Question 11. *Écrire une fonction* Python *qui prend en argument un graphe* G*. Cette fonction doit renvoyer* `True` *si* G *est acyclique, et renvoyer* `False` *sinon.* **On supposera que** G **est connexe. Le nombre d'opérations doit être linéaire en la taille de** G**.** Q.11