Feuille d'exercices n°2 - Graphes et parcours

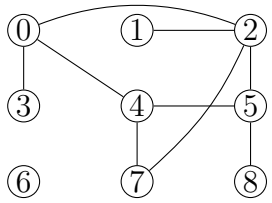
Notions abordées

- Notion de bordure, de parcours (quelconque, largeur, profondeur)
- Tri topologique, tri préfixe
- Forêt associée à un parcours
- Parcours de graphe non orienté pour le calcul des composantes connexes
- Parcours de graphe non orienté pour la détection de graphe biparti
- Algorithme de Dijkstra et $A^\star$

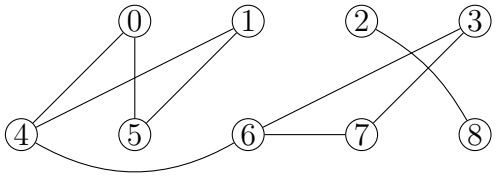
Exercice 1 : Parcours et tri des sommets d'un graphe

Dans tout cet exercice, lorsqu'on appliquera un algorithme du cours, on s'efforcera de l'appliquer en considérant les sommets par numéros croissants, en particulier lorsqu'on visite les voisins d'un sommet. De plus, lorsqu'on donnera un parcours, on soulignera les points de régénération.

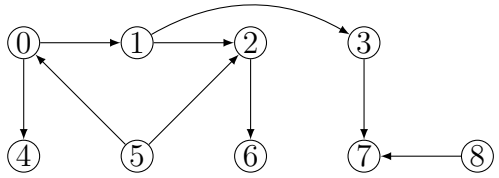
G_1 :



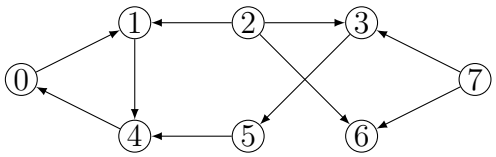
G_3 :



G_4 :



G_5 :



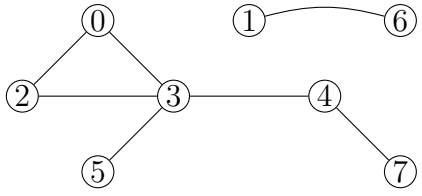
Q. 1 Donner un parcours en largeur et un parcours en profondeur du graphe G_1.
Peut-on en déduire le nombre de composantes connexes de G_1 ?

Q. 2 Justifier pourquoi $[0, 1, 2, 3, 4, 5, 7, 8]$ et $[0, 3, 4, 1, 2, 5, 7, 8, 6]$ ne sont pas des parcours de G_1.

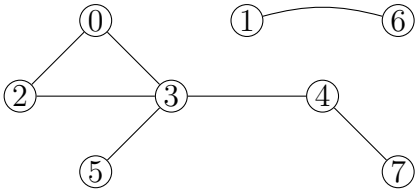
Q. 3 Proposer un parcours des sommets du graphe G_1 qui n'est ni un parcours en profondeur ni un parcours en largeur. Justifier qu'il n'est ni l'un ni l'autre.

Q. 4 Représenter ci-dessous une forêt de parcours en largeur (resp. en profondeur) du graphe G_2. Indiquer le parcours correspondant.

G_2 :



G_2 :



✎ **Q. 5** Donner un parcours en largeur et un parcours en profondeur du graphe G_3. Donner les forêts associées à ces parcours.

Q. 6 Donner un parcours en largeur, un parcours en profondeur et un tri préfixe du graphe G_4.

Q. 7 Donner un tri topologique du graphe G_4 s'il en existe un, justifier qu'il n'en existe pas sinon.

Q. 8 Donner le graphe réduit de G_4.

✎ **Q. 9** Reprendre les questions **Q. 6** à **Q. 8** mais sur le graphe G_5 au lieu de G_4.

Exercice 2 : Calcul des composantes connexes

Dans cet exercice, les graphes non orientés considérés sont de la forme $G = (S, A)$ où $S = [\![0, n[\![$ pour un certain $n \in \mathbb{N}$. On suppose alors qu'ils sont représentés par leur table de listes d'adjacence.

Q. 1 Donner le pseudo-code d'un algorithme permettant le calcul des composantes connexes d'un graphe non orienté. Cet algorithme devra retourner les composantes connexes sous la forme d'un tableau T indicé par $S = [\![0, n[\![$, contenant des entiers de l'intervalle $[\![0, p[\![$ où p est le nombre de composantes connexes du graphe. Cela signifie que pour tout $(u, v) \in [\![0, n[\![^2$, u et v sont dans la même composante connexe de G si et seulement si $T[u] = T[v]$.

Q. 2 Donner la complexité de l'algorithme proposé à la **Q. 1**.

Q. 3 L'algorithme proposé à la **Q. 1** permet-il aussi le calcul des composantes fortement connexes d'un graphe orienté ? Justifier ou donner un contre-exemple.

Q. 4 Donner un algorithme prenant en argument un graphe non orienté $G = (S, A)$ et une liste L de sommets de S, sans redondance, et retournant si oui ou non l'ensemble des sommets de L est connexe. Donner la complexité de cet algorithme.

Exercice 3 : Détection de circuit

Q. 1 Donner le pseudo-code d'un algorithme basé sur un parcours en profondeur qui permet de tester si un graphe orienté est sans circuit.

Q. 2 Préciser l'implémentation de graphe la plus adaptée à cet algorithme. Donner alors la complexité de l'algorithme proposé.

Q. 3 Modifier ou compléter l'algorithme pour qu'il fournisse un circuit s'il y en a un.

Exercice 4 : Détection de graphe biparti

On rappelle qu'un graphe non orienté $G = (V, E)$ est dit **biparti** s'il existe $\{V_1, V_2\}$ une partition de V telle que les arêtes de G ont toutes une extrémité dans V_1 et l'autre dans V_2.

Q. 1 Écrire le pseudo code d'un algorithme de parcours qui teste si un graphe est biparti en essayant de le bi-colorier de manière gloutonne.

Q. 2 Dans le pseudo-code proposé, identifier quelles sont les opérations coûteuses, et en déduire :
- quelle représentation de graphe est la plus adaptée pour cet algorithme (c'est-à-dire celle pour laquelle l'algorithme sera de plus faible complexité) ;
- quelle structure de données est utilisée pour représenter les sommets visités/ouverts/fermés.

Q. 3 Implémenter une fonction `est_biparti` qui teste si un graphe est biparti.

Exercice 5 : Tri topologique

On rappelle qu'une liste L d'éléments $L_1, L_2, \dots L_n$ est un **tri topologique** (des sommets) d'un graphe orienté $G = (S, A)$ ssi $\{L_i \,|\, i \in [\![1, n]\!]\} = S$ et $\forall (i, j) \in [\![1, n]\!]^2,\ i < j \Rightarrow (L_j, L_i) \notin A$. On dit parfois que L est une permutation des sommets dans laquelle aucun sommet n'est placé après l'un de ses successeurs, ou encore dans laquelle chaque sommet est placé avant tout ses successeurs.

Q. 1 Donner un exemple de tri topologique d'un graphe orienté qui n'est pas un parcours.

Q. 2 Donner un exemple de parcours d'un graphe orienté qui n'est pas un tri topologique.

Q. 3 Que peut-on dire du premier sommet d'un tri topologique. Combien a-t-il de prédécesseurs ? de successeurs ? Et le dernier ?

Q. 4 Existe-t-il toujours un tri topologique ? Donner une condition nécéssaire et suffisante à l'existence d'un tri topologique dans un graphe.

Q. 5 Proposer un algorithme en pseudo-code qui permet de calculer un tri topologique s'il en existe un, et qui donne une preuve (au sens témoin, pas au sens démonstration) qu'il n'en existe pas sinon. *On peut essayer de construire le tri topologique pas à pas en remarquant que si $(L_i)_{i \in [\![1,n]\!]}$ est un tri topologique de $G = (S, A)$, alors $(L_i)_{i \in [\![2,n]\!]}$ est un tri topologique du graphe induit par $S \setminus \{L_1\}$.*

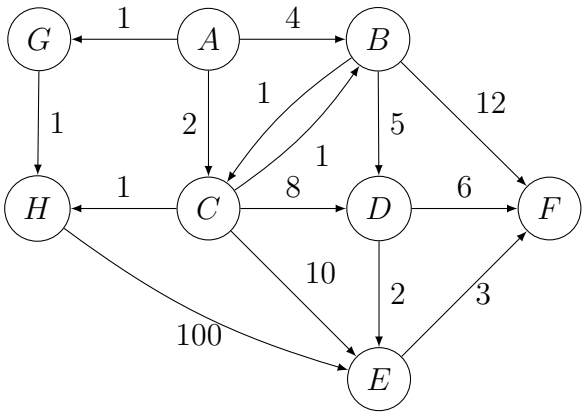
Q. 6 En utilisant la représentation qui permet la complexité la plus intéressante, implémenter l'algorithme proposé. *On attend une complexité pire cas en $O(n + m)$.*

Q. 7 Comment détecter si un graphe orienté présente des circuits.

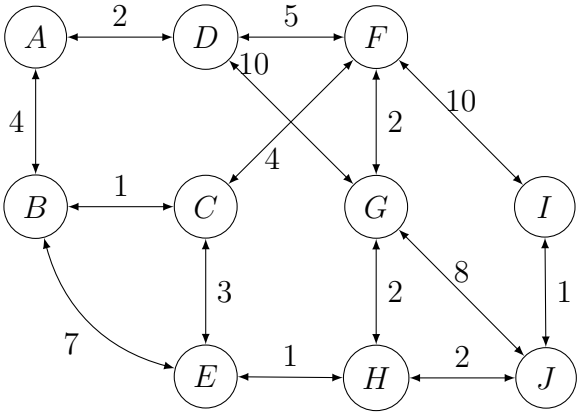
Q. 8 Proposer un algorithme, de complexité $\mathcal{O}(|S| + |A|)$ permettant de calculer un tri topologique dans un graphe orienté $G = (S, A)$.

Exercice 6 : Application de l'algorithme de Dijkstra

Pour cet exercice, on considère les deux graphes ci-dessous.



Graphe G_a



Graphe G_b

Q. 1 Au moyen de l'algorithme de Dijkstra, donner, pour chaque sommet x du graphe G_a, un plus court chemin du sommet A au sommet x.

Q. 2 Au moyen de l'algorithme de Dijkstra, donner, pour chaque sommet x du graphe G_b, un plus court chemin du sommet A au sommet x.

Q. 3 On considère de nouveau le graphe G_a, donner le déroulé de l'algorithme $A^\star$ de recherche d'un plus court chemin de A à E pour l'heuristique h donnée dans le tableau ci-dessous.

x	A	B	C	D	E	F	G	H
$h(x)$	2	4	7	1	0	153	8	0

Q. 4 On considère de nouveau le graphe G_b, donner le déroulé de l'algorithme $A^\star$ de recherche d'un plus court chemin de A à I pour l'heuristique donnée dans le tableau ci-dessous.

x	A	B	C	D	E	F	G	H	I	J
$h(x)$	153	8	7	13	4	8	0	3	0	1