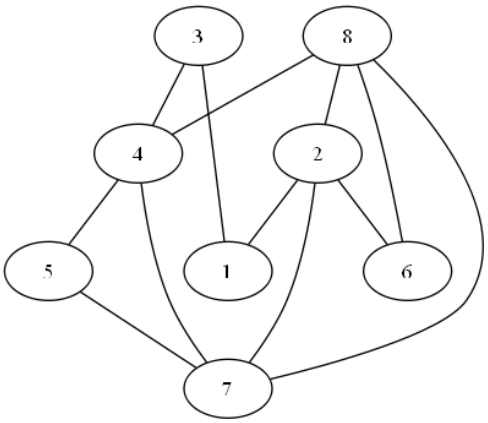
UNIVERSITE DE LA ROCHELLE
Licence Informatique 3ème année
Semestre 5 – *Algorithmique*

TD Graphes - Autour des parcours d'un graphe

K. Bertet – J.L. Guillaume – A. Huchet

Les exercices (au moins jusqu'au 15 inclus) sont à faire en séance, il est fortement conseillé de faire tous les autres à la maison, y compris ceux de la fiche exercices supplémentaires disponible sur moodle.
Les exercices 18 et 19 du sujet en ligne doivent être faits à la maison de manière individuelle et rendus à la deuxième séance de TD (pas de rendu ou absence non justifiée à la seconde séance = 0).

1. Retour sur le cours



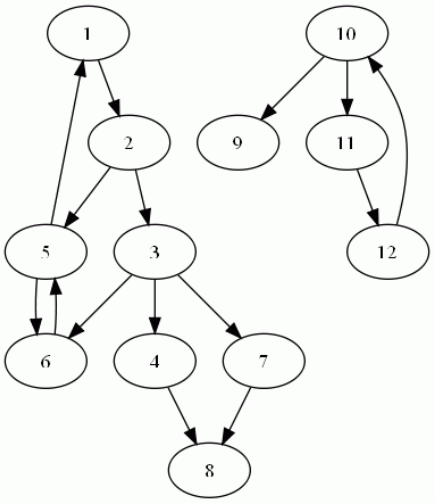
Exercice 1. Le réseau routier modélisé par des sommets pour les intersections et des arêtes pour les routes entre ces intersections est-il orienté ? Peut-il contenir des boucles ? des arêtes multiples ?
Exercice 2. Donner plusieurs exemples de graphes bipartis (où les sommets sont dans deux ensembles et les arêtes uniquement d'un ensemble vers l'autre) modélisant des situations réelles.
Exercice 3. Représenter le graphe ci-contre sous forme de matrice d'adjacence.
Exercice 4. Comment caractériser, à partir de sa matrice d'adjacence :

a) Les boucles d'un graphe ?
b) Un graphe non orienté ?
c) Les sources (sommets sans prédécesseur) et les puits (sommets sans successeur) d'un graphe orienté ?

2. Parcours d'un graphe

Parcours d'un graphe : Méthode d'exploration d'un graphe en parcourant les sommets généralement de proche en proche. Les parcours les plus connus sont les parcours en largeur et en profondeur.

Exercice 5. Appliquer l'algorithme de parcours en largeur pour parcourir le graphe de l'exercice précédent. Les voisins seront toujours traités par ordre croissant.
Exercice 6. Même question avec l'algorithme de parcours en profondeur. Quelles sont les différences entre les arbres obtenus par les deux parcours ?
Exercice 7. Faire un DFS rapide (uniquement pour stocker l'arbre) pour parcourir le graphe ci-contre. Les sommets seront traités par ordre croissant (en commençant par 1). Plusieurs parcours sont nécessaires pour parcourir l'ensemble des sommets de ce graphe. Pourquoi ?
Exercice 8. Combien de parcours seraient nécessaires si le sommet origine était le sommet 4 (ensuite on respecte l'ordre) ? Que préconiseriez-vous pour limiter au maximum le nombre de parcours d'un graphe orienté ?



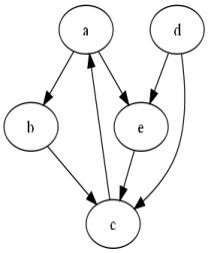
3. Composantes Fortement Connexes

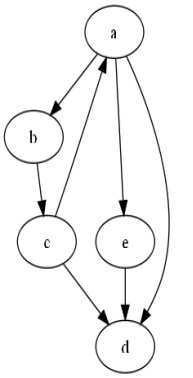
Connexité : Un graphe non orienté est ***connexe*** s'il existe une ***chaîne*** reliant tous les sommets deux à deux. Un sous-graphe (induit) connexe maximal d'un graphe orienté est une ***composante connexe.***

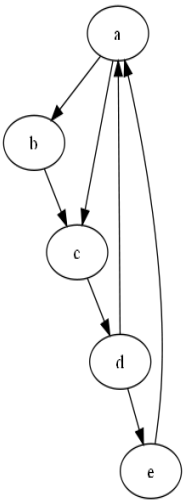
Connexité forte : Un graphe orienté est ***fortement connexe*** s'il existe un ***chemin*** reliant chacun de ses sommets deux à deux. Un sous-graphe induit connexe maximal d'un

graphe orienté est une ***composante fortement connexe***. Un graphe est fortement connexe s'il possède une seule composante.

Algorithme de calcul des composantes fortement connexes :
Entrée : un graphe orienté
Sortie : La liste de ses composantes fortement connexes
1. Faire un DFS en gardant les dates de début et fin de visite.
2. Inverser les arcs du graphe.
3. Refaire un DFS par ordre décroissant de fin de visite.
4. Les arbres issus de ce second parcours sont les composantes fortement connexes.



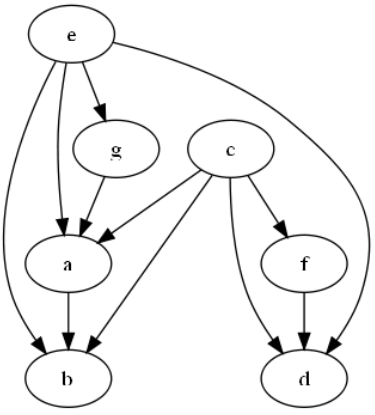




Exercice 9. Parmi les trois graphes ci-contre, lesquels sont fortement connexes ?
Exercice 10. Appliquer l'algorithme sur un de ces graphes et faire les autres plus tard.

4. Tri Topologique

Tri topologique : Se définit pour un graphe orienté sans cycles (DAG = directed acyclic graph). Il s'agit d'un tri des sommets du graphe respectant les contraintes de précédences données par les arcs : s'il y a un arc (x,y) alors x apparait avant y dans l'ordre.



Exercice 11. Les tris suivants des sommets sont-ils des tris topologiques du graphe ci-contre :
a) e-g-a-b-c-f-d
b) c-e-g-a-f-b-d
c) e-c-g-b-a-f-d
Exercice 12. Un graphe peut-il avoir plusieurs tris topologiques ?
Exercice 13. Par quels sommets doit commencer un tri topologique ?

Calcul d'un tri topologique (version 1) : Lister les sommets d'un DAG par ordre décroissant de temps de fin de visite d'un parcours en profondeur donne un tri topologique.

Exercice 14. Appliquer cet algorithme pour le graphe de l'exercice précédent et vérifier le résultat.

Calcul d'un tri topologique (version 2) : Choisir une source s, supprimer s du graphe et recommencer. L'ordre de suppression des sommets est un tri topologique.

Exercice 15. Appliquer l'algorithme de calcul d'un tri topologique pour le graphe de l'exercice précédent en expliquant comment calculer les sources d'un graphe.
Exercice 16. En pratique on ne supprime pas les sources mais on les marque. La recherche et la mise à jour des sources peut se faire en utilisant un tableau *NbPred* qui enregistre pour chaque sommet son nombre de prédécesseurs non encore traités à chaque étape. Une source est donc un sommet dont tous les prédécesseurs ont été marqués. Donner les détails de l'initialisation de *NbPred e*t sa mise à jour à chaque étape.
Exercice 17. Quel serait le résultat de l'algorithme si l'arête (b,e) était ajoutée ? En déduire un algorithme de détection de cycles dans un graphe orienté.

5. Exercices supplémentaires

Exercice 18. Donner une intuition simple d'un algorithme pour compter le nombre de tris topologiques d'un DAG. On ne demande pas de pseudo-code.
Exercice 19. Montrer que G est un DAG alors toutes ses CFC contiennent un seul sommet.
Exercice 20. Ecrire l'algorithme de parcours en profondeur de manière itérative en utilisant une pile (à la place des appels récursif). Pour rappel une pile a trois opérations principales : estVide(p) qui retourne vrai ou faux, empiler(p,x) qui ajoute x en haut de p et ne retourne rien, dépiler(p) qui supprime et retourne l'élément en haut de p. Simuler l'exécution sur un des graphes de l'énoncé en indiquant à chaque fois le contenu de la pile et l'état des sommets.
Exercice 21. On considère le graphe G' des composantes fortement connexes d'un graphe G. Le graphe G' a un sommet pour chaque CFC de G et s'il y a un arc d'une CFC vers une autre dans G alors il y a un arc équivalent dans G'. Donnez le graphe des CFC des graphes de la partie 3. Montrez que le graphe des CFC de tout graphe est un DAG.
Exercice 22. Montrer que si toutes les CFC d'un graphe ne contiennent qu'un seul sommet alors ce graphe est un DAG.
Exercice 23. Compter précisément le nombre d'opérations des différents algorithmes vus jusqu'à présent (on comptera au minimum : les comparaisons, les opérations arithmétiques, les affectations).
Exercice 24. Tester tous les algos sur tous les graphes du sujet (en ajoutant des orientations si nécessaire)
Exercice 25. En pratique dans le tri topologique v2 on ne supprime pas les sources mais on les marque. La recherche et la mise à jour des sources peut se faire en utilisant un tableau *NbPred* qui enregistre pour chaque sommet son nombre de prédécesseurs non encore traités à chaque étape. Une source est donc un sommet dont tous les prédécesseurs ont été marqués. Donner les détails de l'initialisation de *NbPred e*t sa mise à jour à chaque étape.
Exercice 26. Coder tous les algorithmes en python en utilisant une représentation du graphe sous la forme que vous préférez (liste d'arêtes ou d'arcs, matrice d'adjacence, listes d'adjacence). On utilisera au TP1 des objets graphes spécifiques en python qui simplifient le travail.