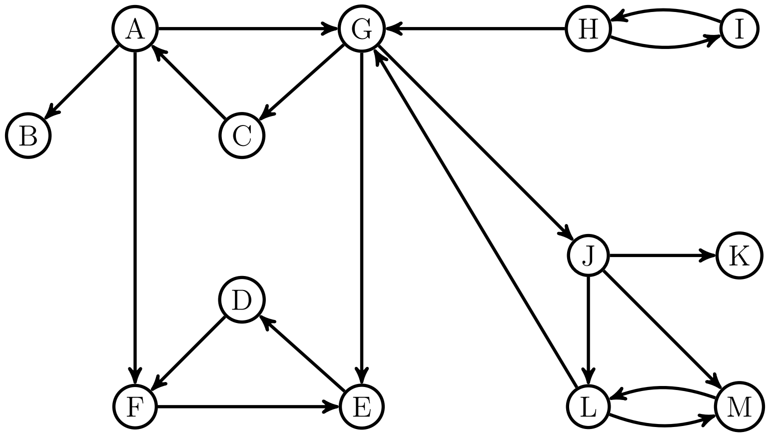
TD 5 - Graphes orientés.

Dans ce qui suit, G désigne un graphe orienté.

Exercice 1.

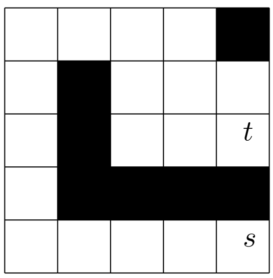
1. Dessinez les *forêts* correspondant à l'exploration en profondeur sur le graphe ci-dessous à partir du sommet A et à partir du sommet H, avec les numérotations correspondantes.



2. Modifiez l'algorithme de parcours en profondeur vu au cours pour qu'il renvoie la forêt d'exploration plutôt que la liste des sommets rencontrés.
3. Modifiez l'algorithme de détection de cycle vu au cours pour qu'il renvoie un cycle s'il en existe un (NIL sinon).
4. En vous inspirant de l'algorithme de parcours en largeur, écrivez un algorithme ACCESSIBLE(G, s, t), qui prend en entrée un graphe orienté G et deux sommets s et t et renvoie VRAI si t est accessible à partir de s dans G (FAUX sinon).
5. Modifiez votre algorithme pour qu'il renvoie un chemin de s à t lorsqu'il en existe un, NIL sinon.

Exercice 2.

On considère un labyrinthe représenté comme une grille $n \times m$ avec des cases blanches parmi lesquelles on dispose d'une case d'entrée s et d'une case de sortie t.



La position d'un personnage dans le labyrinthe est donnée par la case dans laquelle il se trouve ainsi que la direction dans laquelle il regarde. Les règles de déplacement sont les suivantes :

— on ne peut se déplacer que sur les cases blanches (les cases noires sont des murs) ;

— on ne peut changer de case qu'en avançant en ligne droite ou en tournant à droite par rapport à la direction dans laquelle on regarde.

On suppose que le joueur peut choisir son orientation de départ et l'objectif est d'atteindre la case de sortie (peu importe l'orientation à l'arrivée).

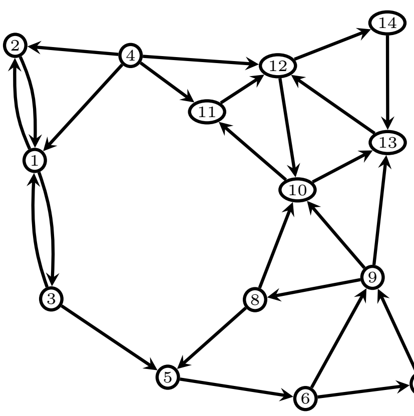
1. Modélisez les chemins possibles dans le labyrinthe sous la forme d'un graphe où les cases s et t seront représentées par un sommet chacun. Indice : utilisez quatre sommets par case blanche du labyrinthe.
2. Écrivez un algorithme qui étant donnés un graphe correspondant à la modélisation d'une grille vide d'un labyrinthe, une liste de cases, et les coordonnées des cases s et t, renvoie le graphe correspondant au labyrinthe dont les murs sont les cases de la liste donnée en entrée.
3. Écrivez un algorithme qui prend en entrée un graphe modélisant un labyrinthe et renvoie VRAI s'il existe un chemin permettant de sortir du labyrinthe et FAUX sinon. (Vous pouvez utiliser les algorithmes vus précédemment à condition de bien préciser leur entrée et leur sortie.)
4. Quelle est la complexité dans le pire cas de l'algorithme en fonction du nombre $|V|$ de sommets et du nombre $|A|$ d'arcs du graphe passé en entrée ?
5. En tenant compte de la forme particulière du graphe passé en entrée, pouvez-vous affiner votre complexité en la donnant en fonction du nombre de sommets du graphe puis en fonction des dimensions $n \times m$ du labyrinthe de départ ?
6. On cherche maintenant, lorsqu'il en existe un, un chemin de s à t qui minimise le nombre de tournants à droite. Comment ajuster votre modélisation pour pouvoir résoudre ce problème ? Quelle est sa complexité ?

Exercice 3.

Composantes fortement connexes (CFC)

Dans un graphe orienté $G = (V, A)$, une composante fortement connexe $\mathcal{C}$ est un sous-ensemble de sommets maximal tel que pour tout $s, t \in \mathcal{C}$ il existe un chemin de s à t.

1. Quelle est la particularité du graphe des composantes fortement connexes (dont les sommets sont les CFC et dont les arcs indiquent quelle CFC est accessible à partir d'une autre) ?
2. Si on ajoute un arc dans un graphe, comment peut évoluer son nombre de composantes fortement connexes ?
3. Réfléchissez à un procédé naïf pour calculer les CFC d'un graphe, sans utiliser l'algorithme du cours.
4. Quelle est la complexité de cet algorithme en fonction de $|V|$ et de $|A|$?
5. Quelle est la complexité de l'algorithme vu au cours pour calculer les composantes fortement connexes ?
6. Appliquez cet algorithme au graphe ci-dessous.



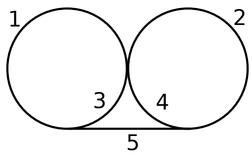
7. Ne pourrait-on pas garder le graphe initial et faire le deuxième parcours suivant les temps de fin de visite croissants ?

Exercice 4.

1. Étant donnés un graphe orienté non pondéré $G = (V, A)$ et un sommet $s \in V$, montrez que l'arbre d'exploration en largeur de G à partir de s est un arbre des plus courts chemins à partir de s.
2. Étant donnés un graphe orienté non pondéré $G = (V, A)$, un sommet $s \in V$ et un arbre des plus courts chemins à partir de s, est-il toujours possible d'obtenir cet arbre par un parcours en largeur de G depuis s ? Prouvez votre affirmation.
3. Vos réponses aux deux premières questions sont-elles toujours valides si G est pondéré ?
4. Peut-on utiliser l'exploration en largeur pour calculer un arbre des plus courts chemins sur un graphe pondéré dont les poids des arcs sont soit 1, soit 2 ?
5. Peut-on étendre cette méthode à un graphe dont les poids des arcs sont quelconques ? Quelle est la complexité dans ce cas-là ?

Exercice 5.

Bob possède un train électrique dont le circuit a la forme suivante. Alice remarque que quel que soit le point d'où il fait démarrer son train et le sens dans lequel il circule, celui-ci pourra rouler indéfiniment mais ne passera jamais plus d'une fois par le tronçon horizontal.



1. Un tronçon du circuit correspond à une portion située entre deux aiguillages. Ils sont notés de 1 à 5 sur la figure ci-dessus. Le sens dans lequel le train circule sur un tronçon a de l'importance pour la suite de son parcours. Par exemple un train circulant sur le tronçon 5 de gauche à droite ne peut pas continuer sur le tronçon 4.

   Modélisez les parcours possibles du train à l'aide d'un graphe orienté dont les sommets représentent chacun un tronçon du circuit avec son sens de parcours.
2. Expliquez/justifiez l'observation d'Alice sur le graphe obtenu.