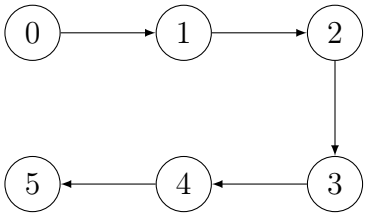
TD 25 : Graphe du métro parisien, Parcours de graphe

Fichiers fournis

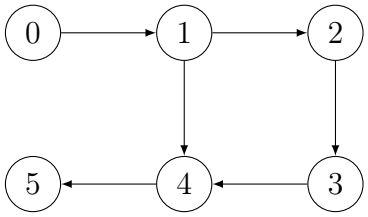
On vous fournit une correction du début du TP précédent, vous permettant d'obtenir une représentation par liste d'adjacence du graphe du métro parisien, à partir du fichier textuel qui le décrit.
Le code du TD précédent a été encapsulé dans une fonction `get_metro_parisien_la`.

Exercice 1 (Cours : Nombre d'arcs entre deux sommets)

Coder une fonction `nombre_etapes(g, source, dest)` qui prend en entrée un graphe orienté `g` représenté par listes d'adjacence, un indice de sommet `source` et un indice de sommet `dest` (comme *destination*) et qui renvoie en sortie le plus petit nombre d'arcs du graphe qu'on peut traverser pour aller de `source` à `dest`.
(Remarque : nous n'avons pas appelée cette fonction `distance`, car elle ne prend pas en compte le fait que certains arcs prennent plus de temps à être empruntés que d'autres.)



Le plus court chemin de `0` à `5` est de longueur 5.

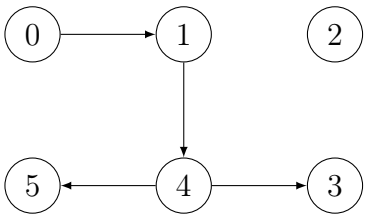


Le plus court chemin de `0` à `5` est de longueur 3 à cause du raccourci entre `1` et `4`.

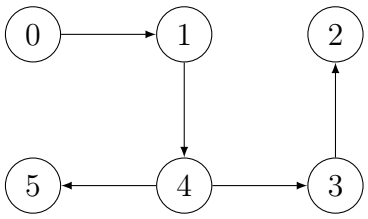
Exercice 2

On cherche à vérifier qu'à partir d'une station de métro donnée on peut accéder à n'importe quelle autre station. Pour cela, une façon de faire est de lancer un parcours en profondeur depuis la station de départ, puis vérifier qu'à la fin du parcours toutes les stations du graphe ont été visitées.

1. Coder une fonction `donne_acces_a_tout(g, s)` qui prend en entrée un graphe orienté `g` représenté par listes d'adjacence et un indice de sommet `s` et renvoie `True` si on peut aller depuis `s` vers tous les autres sommets du graphe, et `False` sinon.
2. Coder une fonction `fortement_connexe(g)` qui prend en entrée un graphe orienté `g` représenté par listes d'adjacence et qui renvoie `True` si depuis **n'importe quel** sommet de `g` on peut accéder à n'importe quel autre sommet de `g`. (Votre fonction pourra appeler la fonction `donne_acces_a_tout`).



Le sommet `2` ne peut être atteint depuis aucun autre sommet.



Le sommet `2` est maintenant atteignable via le sommet `3`.

Exercice 3 (Bonus : composantes fortement connexes)

1. Écrire une fonction `effondrement(g, p)` qui prend en entrée un graphe `g` et un flottant `p` entre 0 et 1, et renvoie une copie du graphe `g` dans laquelle une part `p` des arêtes (choisies au hasard) ont été retirées. Si m est la taille (nombre d'arêtes) du graphe en entrée, le résultat sera en moyenne de taille $p \times m$.
2. Dans un graphe orienté $G = (S, A)$, on appelle *composante fortement connexe* du graphe toute partie $C \subseteq S$ telle qu'il existe un chemin entre n'importe quelle paire de sommets de C, et qui soit maximale par l'inclusion parmi ces parties-là.
   Écrire une fonction `composantes_fortement_connexes(g)` qui prend en entrée un graphe et renvoie en sortie une liste qui à chaque indice de sommet associe un indice de composante fortement connexe à laquelle il appartient. (Si il y a k composante fortement connexes dans les graphes, on les numérotera de 0 à $k-1$. On est libre de choisir n'importe quelle numérotation possible.)
   (Si besoin d'aide, chercher sur internet l'algorithme de Kosaraju-Sharir).
3. Quelle est la complexité de votre fonction ?
4. Tracer une courbe qui à différentes valeurs possibles de `p` associe le nombre de composantes fortement connexes dans `effondrement(metro_la, p)`.
   (Revoir le TD 11 si besoin).