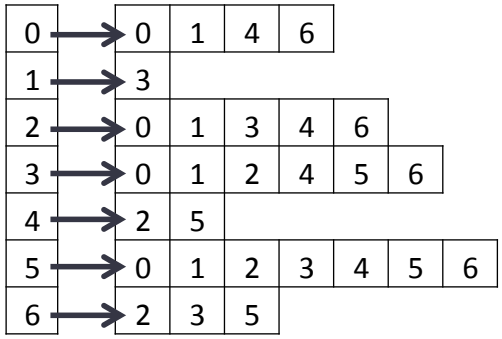
Représentation d'un graphe

- Représentation par liste d'adjacence
  - Soit un graphe $G = (V,E)$ d'ordre n
    - Tel que $V = (0,1,2,3,\ldots,n\text{-}1)$ *i.e.* les sommet sont numérotés de 0 à n-1
  - On le représente avec une liste L de taille n, dont chaque élément est une liste
    - Chacune de ces « sous-listes » contient au plus n éléments, chaque élément étant le numéro d'un sommet
  - Cette liste $L = [L_0, L_1, L_2, L_3, \ldots, L_{n-1}]$, est construite telle que
    - Les rangs de la liste L décrivent tous les sommets $\in V$
    - Les éléments des listes L_i sont des sommets $\in V$
    - Le sommet j apparaît dans la liste L_i si $(i,j) \in E$
  - Dans le cas d'un graphe orienté, les rangs de la liste L décrivent les sommets initiaux et les éléments des listes L_i les sommets terminaux

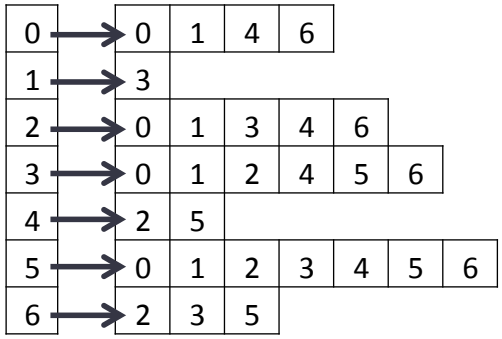
Représentation d'un graphe

- Représentation par liste d'adjacence
  - Soit un graphe $G = (V,E)$ d'ordre n
    - Tel que $V = (0,1,2,3,\ldots,n-1)$ *i.e.* les sommet sont numérotés de 0 à n-1
  - On le représente avec une liste de taille n, dont chaque élément est une liste
    - Chacune de ces « sous-listes » contient au plus n éléments, chaque élément étant le numéro d'un sommet
  - Par exemple, pour le graphe graphe orienté $G_o = (V_o,E_o)$ on a
    - $L_0 =$



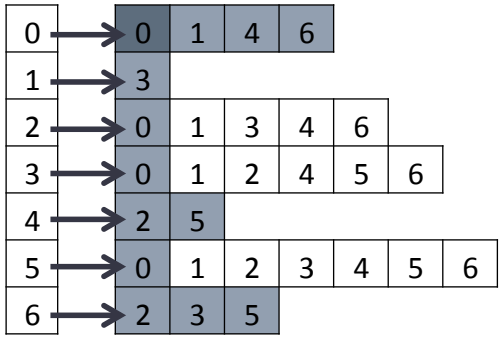
Représentation d'un graphe

- Représentation par liste d'adjacence
  - Soit un graphe $G = (V,E)$ d'ordre n
    - Tel que $V = (0,1,2,3,\ldots,n-1)$ *i.e.* les sommet sont numérotés de 0 à n-1
  - On le représente avec une liste de taille n, dont chaque élément est une liste
    - Chacune de ces « sous-listes » contient au plus n éléments, chaque élément étant le numéro d'un sommet
  - Par exemple, pour le graphe graphe orienté $G_o = (V_o,E_o)$ on a
    - $L_0 =$



Représentation d'un graphe

- Représentation par liste d'adjacence
  - Étant donné un graphe *G* décrit par sa liste d'adjacence *L*
    - Pour savoir si l'arc ou l'arrête (*i,j*) existe dans *G,* il faut consulter tous les éléments de la liste L_i
    - Pour connaître le voisinage ou le degré d'un sommet, il faut, dans le cas d'un graphe orienté, consulter tous les éléments de toutes les sous-listes
  - Par exemple, pour le graphe graphe orienté $G_o = (V_o, E_o)$ on a
    - **N.B.** On note que les sous-listes sont ordonnées pour cet exemple, ce qui ne fait pas partie de la définition de base d'une liste d'adjacence. Cela réduit le coût de la recherche, mais induit un surcoût pour trier les listes au préalable.
    - L_0 =



voisinage-sortant(0) = L_0 = {0,1,4,6}

voisinage-entrant(0) = {0,2,3,5}

Représentation d'un graphe

- Avantages et inconvénients des deux types de représentations
  - Matrice d'adjacence
    - La quantité de mémoire nécessaire est liée au nombre de sommets : n^2
    - Le test de l'existence d'une arrête/d'un arc est immédiat
    - Pour connaître le voisinage/degré d'un sommet, il faut parcourir une ligne et/ou une colonne
  - Liste d'adjacence
    - Quantité de mémoire nécessaire liée à $|E|$, *i.e.* le nombre de liens : $n + m$
    - Le test de l'existence d'une arrête/arc requiert de parcourir des listes
    - Pour connaître le voisinage/degré d'un sommet, il suffit dans certains cas de récupérer une liste
  - Le choix du mode de représentation d'un graphe dépend notamment de la **densité** de ce graphe

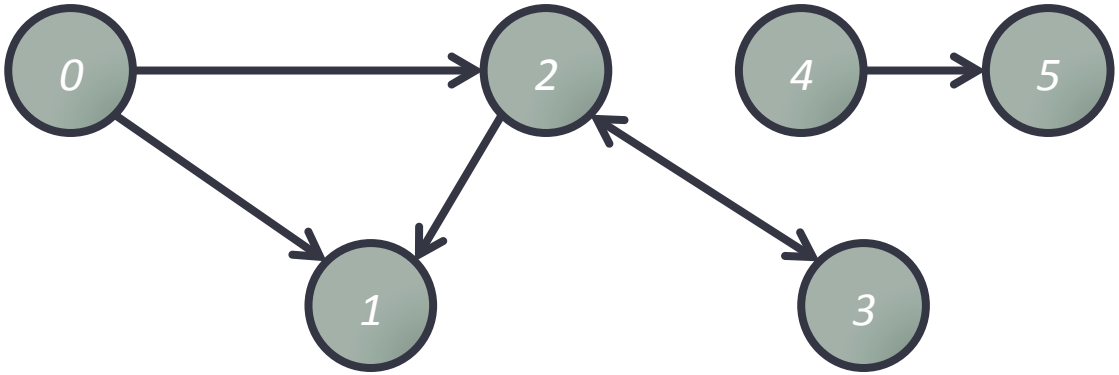
Représentation d'un graphe

- Densité d'un graphe
  - Ratio entre la nombre de liens d'un graphe ($|E|$) et le nombre de liens possibles au sein de ce graphe (*e.g.* n^2 pour un graphe orienté, ½ x n x (n – 1) + n pour un graphe non orienté)

- Avantages et inconvénients des deux types de représentations
  - Matrice d'adjacence
    - Plutôt employée lorsque la graphe est dense
  - Liste d'adjacence
    - Plutôt employée lorsque le graphe est peu dense, *i.e.* creux ($|E| \ll n^2$)
  - Le choix du mode de représentation d'un graphe dépend notamment de la densité de ce graphe

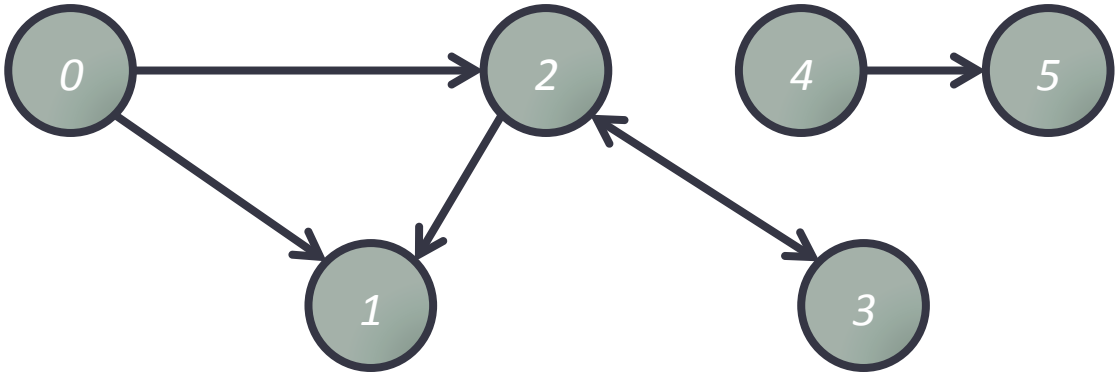
Exercices (I)

- Donnez les représentations par matrice d'adjacence et liste d'adjacence du graphe dessiné ci-dessous



Exercices (I)

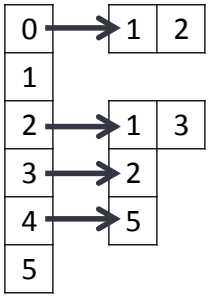
- Donnez les représentations par matrice d'adjacence et liste d'adjacence du graphe dessiné ci-dessous



$A =$

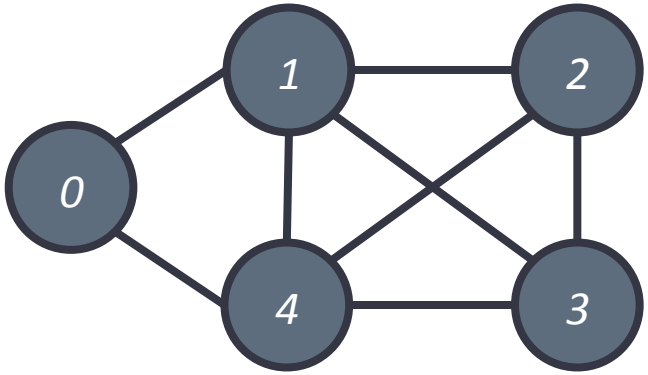
	0	1	2	3	4	5
0	0	1	1	0	0	0
1	0	0	0	0	0	0
2	0	1	0	1	0	0
3	0	0	1	0	0	0
4	0	0	0	0	0	1
5	0	0	0	0	0	0

$L =$



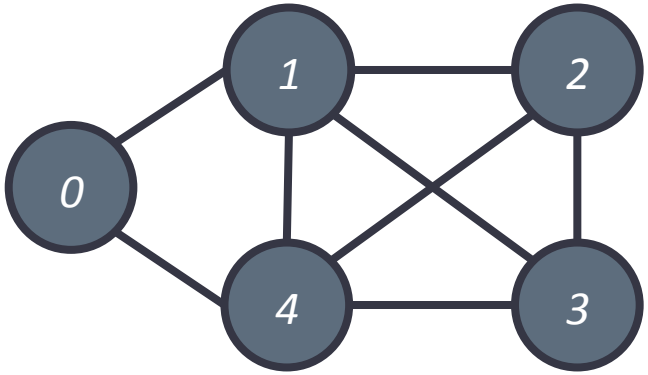
Exercices (II)

- Donnez les représentations par matrice d'adjacence et liste d'adjacence du graphe dessiné ci-dessous



Exercices (II)

- Donnez les représentations par matrice d'adjacence et liste d'adjacence du graphe dessiné ci-dessous



$A =$

	0	1	2	3	4
0	0	1	0	0	1
1	0	0	1	1	1
2	0	0	0	1	1
3	0	0	0	0	1
4	0	0	0	0	0

$L =$

0	1	4	
1	2	3	4
2	3	4	
3	4		
4			

Exercice (III)

- Concevez un programme écrit en Python permettant, étant donné une matrice d'adjacence, de connaître l'ordre du graphe, le nombre de liens dans le graphe, le degré d'un sommet (degré entrant et degré sortant le cas échéant) et son voisinage (même remarque)
  - Partez du fichier TP1.py disponible sur le bureau virtuel
  - Écrivez trois fonctions sans paramètre
    - ordre(), nombreDeLiens() et densité()
  - Écrivez six fonctions prenant un sommet en paramètre
    - degréSortant(sommet), degréEntrant(sommet) et degré(sommet)
    - voisinageSortant(sommet), voisinageEntrant(sommet) et voisinage(sommet)
  - Affichez les propriétés du graphe manipulé
  - Calculez le degré maximum et le degré minimum du graphe
  - Faites en sorte que l'utilisateur puisse connaître le degré, voisinage, *etc.* d'un sommet de son choix