Graphes

On ne considère dans cette feuille de révisions que des graphes :

- non orientés ;
- sans boucle (un sommet n'est jamais relié à lui-même par une arête) ;
- simples (pas plus d'une arête entre deux sommets) ;
- à n sommets, numérotés de 0 à $n-1$.

I. Matrice d'adjacence et liste des listes d'adjacence

Exercice 1

1. Compléter la fonction **Python** qui suit pour qu'elle prenne en entrée la matrice d'adjacence d'un graphe G et qu'elle renvoie la liste des listes d'adjacence de ce graphe.

```
1  def matrice2listeAdj(M):
2      n = __len(M)__
3      L = [[] for k in range(n)]
4      for i in range(n):
5          for j in range(i+1,n):
6              if __M[i,j] == 1__: # Si i et j sont voisins
7                  L[i].append(j)
8                  L[j].append(i)
9      return L
```

Commentaire

Soit G un graphe. Soit x un sommet de G. On appelle *liste d'adjacence* du sommet x la liste des voisins de x. Par extension, la liste des listes d'adjacence de G est la liste de toutes les listes d'adjacence des sommets de G.
Dans le programme précédent, on utilise le fait que G est non orienté donc la matrice d'adjacence est symétrique. Ceci a permis de ne visiter que la partie triangulaire supérieure de la matrice (économie de temps).

2. Compléter la fonction **Python** qui suit pour qu'elle prenne en entrée la liste des listes d'adjacence d'un graphe G et qu'elle renvoie la matrice d'adjacence de ce graphe.

```
1  def liste2matriceAdj(L):
2      n = __len(L)__
3      M = np.zeros([n,n])
4      for i in range(n):
5          for j in __L[i]__:
6              M[i,j] = __1__
7      return M
```

II. Degré d'un sommet

On rappelle que le degré d'un sommet i est le nombre de voisins de i, c'est-à-dire le nombre de sommets du graphe qui sont reliés à i par une arête.

Exercice 2
On pourra utiliser les fonctions **Python** dans les questions qui suivent.

1. a) Compléter la fonction **Python** qui suit pour qu'elle prenne en entrée la matrice d'adjacence d'un graphe G ainsi que le numéro d'un sommet i et qu'elle renvoie le degré du sommet i.

```
1  def degSommetMat(M,i):
2      return __np.sum(M[i])__
```

b) Compléter la fonction **Python** qui suit pour qu'elle prenne en entrée la liste des listes d'adjacence d'un graphe G ainsi que le numéro d'un sommet i et qu'elle renvoie le degré du sommet i.

```
1  def degSommetListe(L,i):
2      return __len(L[i])__
```

2. a) Compléter la fonction **Python** qui suit pour qu'elle prenne en entrée la matrice d'adjacence d'un graphe G et qu'elle renvoie la liste des degrés des sommets de G.

```
1  def matrice2listeDeg(M):
2      n = __len(M)__
3      D = []
4      for i in range(n):
5          __D.append(np.sum(M[i]))__
6      return D
```

b) Compléter la fonction **Python** qui suit pour qu'elle prenne en entrée la liste des listes d'adjacence d'un graphe G et qu'elle renvoie la liste des degrés des sommets de G.

```
1  def listeAdj2listeDeg(L):
2      n = __len(L)__
3      D = []
4      for i in range(n):
5          __D.append(len(L[i]))__
6      return D
```

3. a) Compléter la fonction **Python** qui suit pour qu'elle prenne en entrée la matrice d'adjacence d'un graphe G et qu'elle renvoie le degré maximal d'un sommet de G.

```
1  def matrice2degMax(M):
2      return __max(matrice2listeDeg(M))__
```

b) Compléter la fonction **Python** qui suit pour qu'elle prenne en entrée la liste des listes d'adjacence d'un graphe G et qu'elle renvoie la liste des degrés des sommets de G.

```
1  def listeAdj2degMax(L):
2      return __max(listeAdj2listeDeg(L))__
```