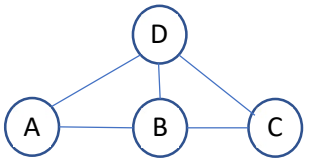
1.3 Listes d'adjacence par listes

Le graphe ci-contre est considéré.



Q1. Écrire les listes d'adjacence par listes. Le graphe sera consigné dans une liste. Pour chacune des listes d'adjacences, les sommets seront des chaines de caractères et chacune des listes d'adjacence sera une liste.

```
graphe=[["A",['B','D']],
        ["B",['A','C','D']],
        ["C",['B','D']],
        ["D",['A','B','C']]]
```

Q2. Écrire une fonction `liste_sommets` qui prend en argument un graphe `g` sous forme définie précédemment et qui renvoie la liste des sommets de ce graphe.

```
def liste_sommets(g):
    L_sommets=[]
    for element in g:
        L_sommets.append(element[0])
    return L_sommets
```

Q3. Écrire une fonction `ajout_sommet_isole` qui prend en argument un graphe `g` sous forme définie précédemment et un point `s` et qui ajoute ce point `s` comme sommet isolé au graphe.

```
def ajout_sommet_isole(g,s):
    g.append([s,[]])
    return g
```

Q4. Écrire une fonction `ajout_arete` qui prend en argument un graphe `g` sous forme définie précédemment et une arête `a` et qui ajoute l'arête au graphe. Une arête est représentée par une liste de deux sommets appartenant à `g` : `[s1,s2]`.

```
def ajout_arete(g,L):
    for i in range(len(g)):
        if g[i][0]==L[0]:
            g[i][1].append(L[1])
        if g[i][0]==L[1]:
            g[i][1].append(L[0])
    return g
```