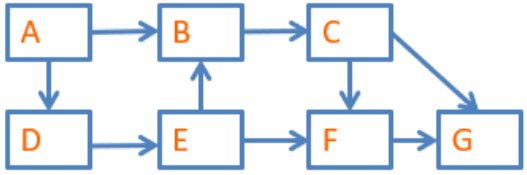
Exercice 5: parcours en largeur d'un graphe

On considère le graphe ci-dessous:



5.1 Coder le graphe

```
g={'A':{'B','D'},'B':{'C'},'C':{'F'},'D':{'E'},.., 'G':set() }
```

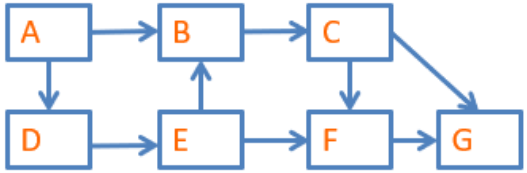
5.2 On cherche à définir le dictionnaire distances qui contient la distance minimale entre un des sommets et tous les autres.

Définir manuellement ce dictionnaire pour le sommet B

```
distances = {'B':0,'C':1,'F':2 ,'G':2}
```

Exercice 5: parcours en largeur d'un graphe

On considère le graphe ci-dessous:




```
def parcours_largeur(g, source):
    """parcours en largeur depuis le sommet source"""
    dist = {source: 0} # un dictionnaire
    courant = { source } # un ensemble
    suivant = set() # un ensemble vide
    while len(courant) > 0:
        s = courant.pop()
        for v in g.voisins(s):
            if v not in dist:
                suivant.add(v)
                dist[v] = dist[s] + 1
        if len(courant) == 0:
            courant, suivant = suivant, set()
    return dist
```

Stratégie

on passe à la fonction le graphe g et un sommet de départ (source)

Dans la fonction:

On crée un dictionnaire dist de distance avec source à distance 0

On crée l'ensemble courant des sommets à analyser (au début contient source)

On crée un ensemble suivant vide pour les voisins du sommet en cours d'analyse

Tant qu'il y a un sommet à analyser dans l'ensemble courant:

on retire un sommet s de l'ensemble source

pour chaque voisin de s, s'il n'est pas déjà dans dist, on l'ajoute dans dist avec distance + 1 par rapport à s et on l'ajoute aussi dans l'ensemble suivant.

si l'ensemble courant est vide, on copie suivant dans courant et on recrée un ensemble suivant vide