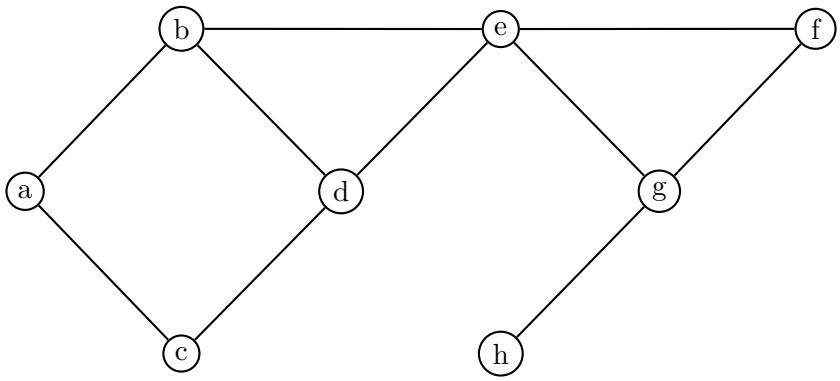
I Parcours en largeur

Parcourir un graphe en largeur à partir d'un sommet, consiste à visiter le sommet puis ses enfants, puis les enfants de ses enfants ...

Comme on l'a déjà vu avec les arbres, il faut utiliser une file et une liste pour marquer les sommets visités.

Prenons en exemple ce graphe :



On souhaite établir un parcours en largeur en partant du sommet `'b'`.

Pour cela, on propose l'algorithme suivant :

```
f =File()
f.enfiler('b')
sommets_visités=[]
Tant que f n'est pas vide :
  s=f.defiler()
  Si s n'est pas dans sommets_visités :
    Ajouter s dans sommets_visités
  Fin si
  Pour chaque voisin de s :
    Si voisin n'est pas dans sommets_visités et n'est pas dans f :
      f.enfiler(voisin)
    Fin si
  Fin pour
Fin tant que
Renvoyer sommets_visités
```

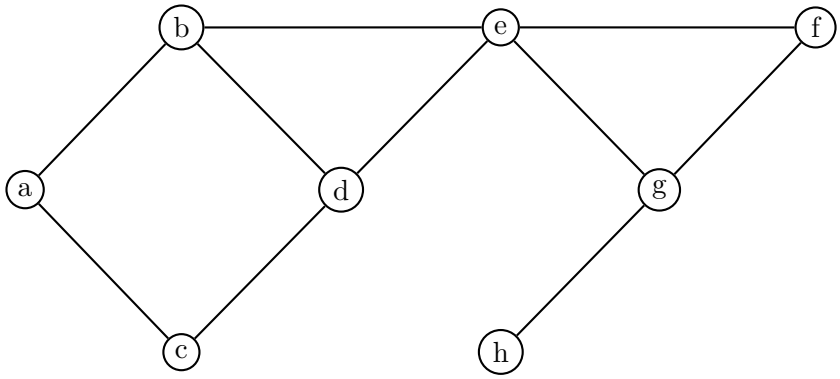
Voici les contenus des variables aux deux premiers tours de la boucle tant que :

- Au premier tour :
  `s = 'b'`
  `sommets_visités = ['b']`
  `f = ['e' , 'd' , 'a']`
- Au deuxième tour :
  `s = 'a'`
  `sommets_visités = ['b', 'a']`
  `f = ['c' , 'e' , 'd']`

Exercice 1 : Compléter les contenus des variables `s`, `sommets_visités` et `f` aux tours suivants puis donner la liste du parcours en largeur de ce graphe.

II Parcours en profondeur

Parcourir un graphe en profondeur à partir d'un sommet, consiste à explorer le graphe en suivant un chemin.
Lorsqu'on arrive sur un sommet qui n'a plus de voisins non visités, on le marque.
Puis on remonte dans le chemin pour explorer les voisins non visités d'un autre sommet.
On utilise une pile `p` et deux listes `sommets_visités` et `sommets_retenus`.
Prenons en exemple le graphe précédent :



On souhaite établir un parcours en profondeur en partant du sommet `'g'`.
Pour cela, on propose l'algorithme suivant :

```
sommets_visités = ['g']
sommets_retenus = []
p = Pile()
p.empiler('g')
Tant que p n'est pas vide :
  s=p.sommet()
  voisins = liste des voisins de s non déjà visités
  Si voisins n'est pas vide :
    v = élément au hasard de voisins
    Ajouter v dans sommets_visités
    p.empiler(v)
  Sinon :
    Ajouter s dans sommets_retenus
    p.depiler()
  Fin si
Fin tant que
Renvoyer sommets_visités
```

Voici les contenus des variables au premier tour de la boucle `tant que` :

```
s : 'g'
voisins : ['e', 'f', 'h']
v : 'e'
sommets_visités : ['g', 'e']
p : ['g', 'e']
sommets_retenus : []
```

Voici les contenus des variables au second tour de la boucle `tant que` :

```
s : 'e'
voisins : ['b', 'd', 'f']
v : 'b'
sommets_visités : ['g', 'e', 'b']
```

```
p : ['g', 'e', 'b']
sommets_retenus : []
```

Exercice 2 : Compléter les contenus des variables aux tours suivants puis donner la liste du parcours en profondeur de ce graphe.