

1. Proposer un parcours en profondeur possible à partir du sommet A.
2. Voici un dictionnaire permettant de définir le graphe :

```
graphhe_oriente = { 'A' : [ 'B' , 'G' ] ,
                     'B' : [ 'C' , 'E' ] ,
                     'C' : [] ,
                     'D' : [ 'F' , 'G' ] ,
                     'E' : [ 'A' , 'D' , 'H' ] ,
                     'F' : [ 'B' , 'C' ] ,
                     'H' : [ 'F' ]
                 }
```

Utiliser l'algorithme de parcours en profondeur sur `graphhe_oriente`.

## 13.3

### Applications

#### 13.3.1 Recherche de chemin

##### Exercice 13.6

Créer une fonction python récursive `parcours_profondeur(graph,sommet,vus = None)` où `graph` est un graphe, `sommet` un des sommets (le sommet de départ) et `vus` un tableau contenant les sommets visités. La fonction renvoie la liste des sommets visités.

En déduire une fonction `existe_chemin(graph,s1,s2)` qui détermine s'il existe un chemin de `s1` à `s2`.

### 13.3.2 Recherche de cycle

Le parcours en profondeur est particulièrement bien adapté à la recherche de cycles dans un graphe.

Le marquage "simple" (tout ou rien) opéré dans l'algorithme précédent ne permet pas de savoir si, lorsqu'on atteint un sommet "visité", il s'agit de la fin d'un parcours.

On utilise donc un marquage des sommets à trois niveaux ("couleurs") :

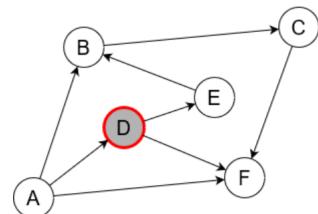
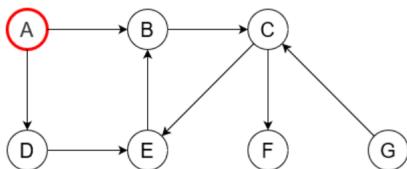
BLANC : non visité , GRIS : visité une seule fois , NOIR : visité deux fois .

Algorithme :

- Tous les sommets sont initialement colorés en BLANC.
- Lorsqu'on visite un sommet :
  - s'il est BLANC :
    - \* on le marque en GRIS (ce sommet a été visité une fois) pour chacun de ses voisins :
    - \* on lance un parcours récursivement si le retour est VRAI (on a déjà découvert un cycle sur un parcours passant par ce sommet) → VRAI
    - \* on le marque en NOIR (le parcours depuis ce sommet n'a pas révélé de cycle)
  - s'il est GRIS (on a découvert un cycle!) → VRAI
  - s'il est NOIR (il n'y a pas de cycle à partir de ce sommet) → FAUX

#### Exercice 13.7

Utiliser les graphes ci-dessous pour analyser l'algorithme ci-dessus.



#### Exercice 13.8

Implémenter une recherche de cycle en python, en créant une fonction `parcours_cycle_r(self, s, coul = None)`. `couleur` est un dictionnaire donnant l'une des 3 couleurs du sommet. On donne le début du programme :

```

def parcours_cycle_r(graph, s, coul = None):
    if coul is None:
        coul = {}
    vus = parcours_prof(graph, s, vus = None) # on colore
    tous les sommets en BLANC
    for s in vus:
  
```

```
coul[s] = 1
```