

# 13.2

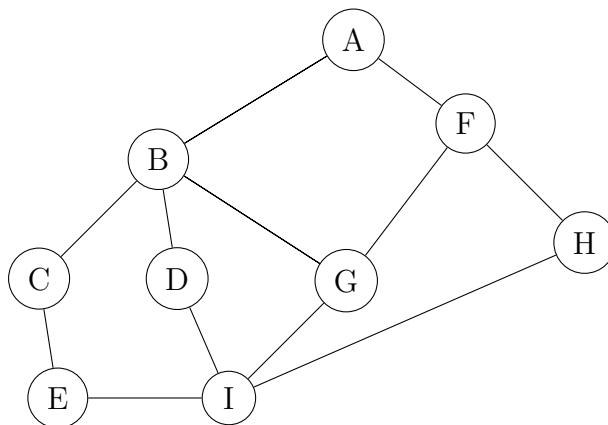
## le parcours en profondeur d'abord

NSI TERMINALE - JB DUTHOIT

```

1 VARIABLE
2 G : un graphe
3 u : noeud
4 v : noeud
5 f : file (initialement vide)
6 # On part du principe que pour tout sommet est initialement vert
7 DEBUT
8 Fonction PARCOURS_PROFONDEUR(G,u)
9   u.couleur ← rouge
10  POUR chaque sommet v adjacent à u FAIRE
11    SI v couleur n'est pas rouge ALORS
12      |  PARCOURS_PROFONDEUR(G,v)

```



**Exercice 13.3**

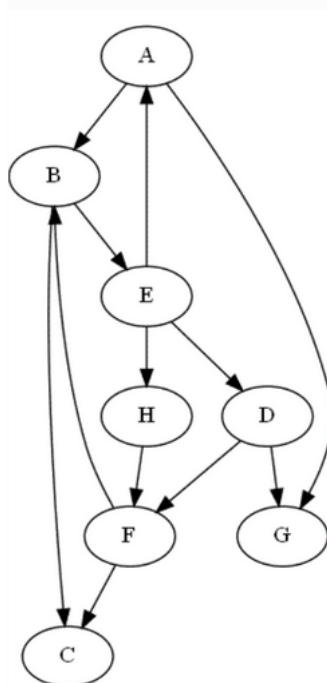
Utiliser l'algorithme du parcours en profondeur d'abord pour ce graphe, et donner l'ordre des sommets visités, en sachant qu'il n'y a pas qu'une seule solution.

**Exercice 13.4**

Implémenter cet algorithme en Python

**Exercice 13.5**

On considère le graphe orienté suivant :



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`.