12.3 Les parcours de graphes

NSI terminale - JB Duthoit

Il existe 2 méthodes pour parcourir un graphe :

- Le parcours en largeur d'abord
- Le parcours en profondeur d'abord

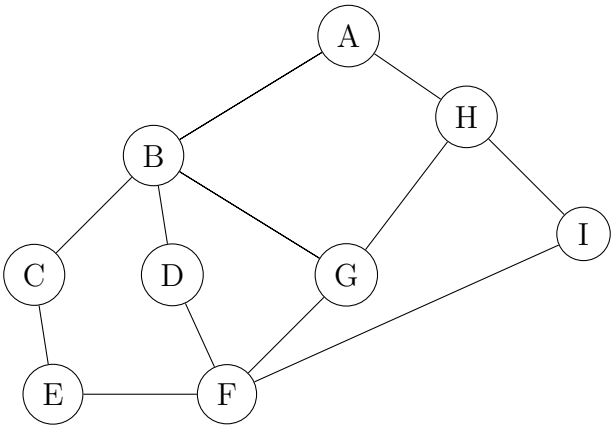
12.3.1 Le parcours en largeur d'abord

Nous allons travailler sur un graphe G(V,E) avec V l'ensemble des sommets de ce graphe et E l'ensemble des arêtes de ce graphe.

On adoptera un code couleur :

- Sommet de couleur verte si le sommet n'a pas encore été visité
- Sommet de couleur rouge si le sommet a été visité

```
1  VARIABLE
2  G : un graphe
3  s : sommet(origine)
4  u : sommet
5  v : sommet
6  f : file (initialement vide)
7  # On part du principe que pour tout sommet est initialement vert
8  Fonction PARCOURS_ LARGEUR(G,s)
9      s.couleur ← rouge
10     enfiler (s,f)
11     TANT QUE f non vide FAIRE
12         u ← defiler(f)
13         POUR chaque sommet v adjacent au sommet u FAIRE
14             SI v.couleur n'est pas rouge ALORS
15                 v.couleur ← rouge
16                 enfiler(v,f)
```



Remarque

- On commence à visiter A

Exercice 12.11

Utiliser l'algorithme du parcours en largeur d'abord pour ce graphe, et donner l'ordre des sommets visités.
On pourra compléter un tableau de ce type :

Etapes	traintement	état de la file à la fin de l'étape
1	A	—A—>
2	f non vide -> u=A (on défile A) -> v=B non rouge donc B + enfile(B) -> v=H non rouge donc H + enfile(H)	—HB—>
3	f non vide -> u=B (on défile B) -> v=A rouge -> v=C non rouge donc C + enfile(C) -> v=D non rouge donc D + enfile(D) -> v=G non rouge donc G + enfile(G)	—GDCH—>
4		
5		
6		
7		
8		
9		
10		

***

Exercice 12.12

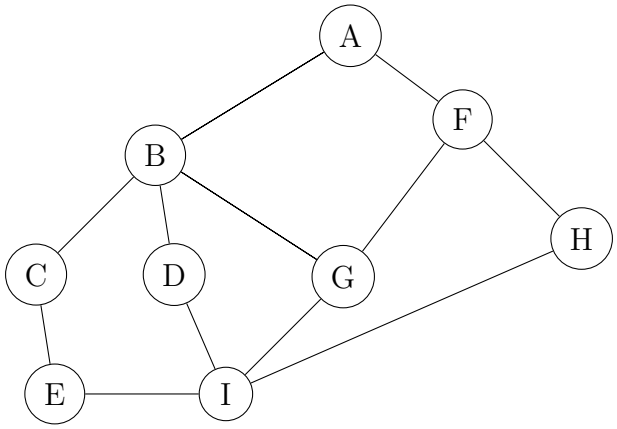
Donner un ordre possible d'affichage des sommets si on effectue un parcours en largeur d'abord à partir du sommet C.

Exercice 12.13

1. Créer une classe `file`, avec les méthode `enfiler`, `défiler` et `est_vide`.
2. Reprendre la classe `GrapheNO`
3. Implémenter cet algorithme (on pourra insérer un `print` au moment où le sommet passe au rouge)

12.3.2 le parcours en profondeur d'abord

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



Exercice 12.14

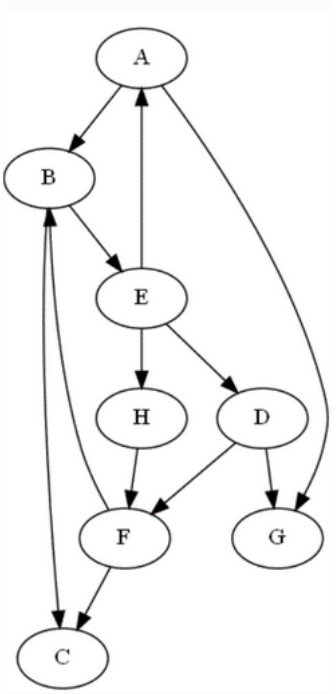
Utiliser l'algorithme du parcours en profondeur d'abord pour ce graphe, et donner l'ordre des sommets visités, en sachant que l'on débute par le sommet 'A', et qu'il n'y a pas qu'une seule solution.

Exercice 12.15

Implémenter cet algorithme en Python

Exercice 12.16

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 :

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

   a) Utiliser l'algorithme du parcours en profondeur d'abord pour ce graphe, et donner l'ordre des sommets visités, en sachant que l'on débute par le sommet 'A', et qu'il n'y a pas qu'une seule solution.

   b) Créer une fonction qui parcourt en profondeur sur `graphe_oriente`, et vérifier les résultats.