

13.1

Algorithme sur les 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

13.1.1 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.

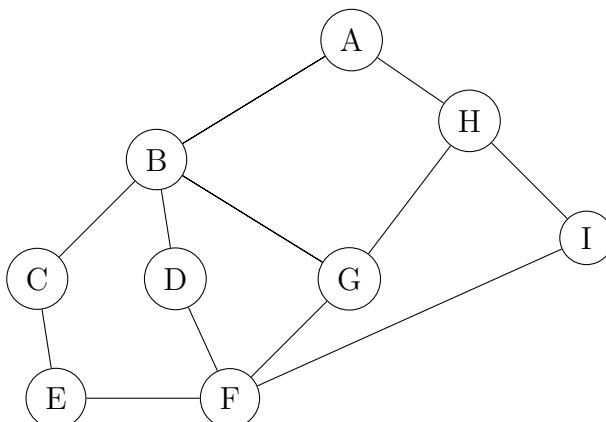
On adoptera un code couleur :

- 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 13.1

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	traitement	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 13.2

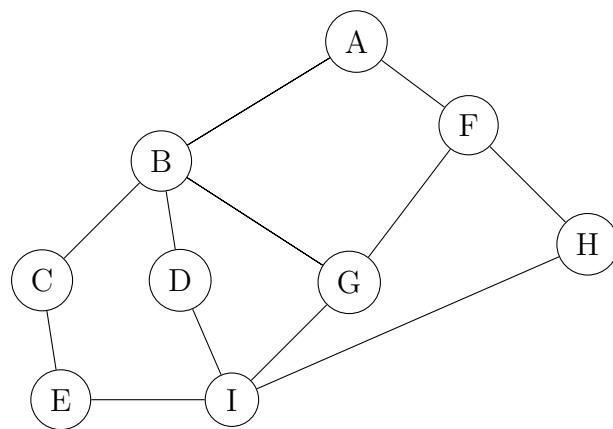
1. Créer une classe `file`, avec les méthodes `enfiler`, `défiler` et `est_vide`.
2. Reprendre la classe `GrapheNO`
3. Implémenter cet algorithme (on pourra insérer un `print` au moment où le sommet passe au rouge)

13.1.2 le parcours en profondeur d'abord

```

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

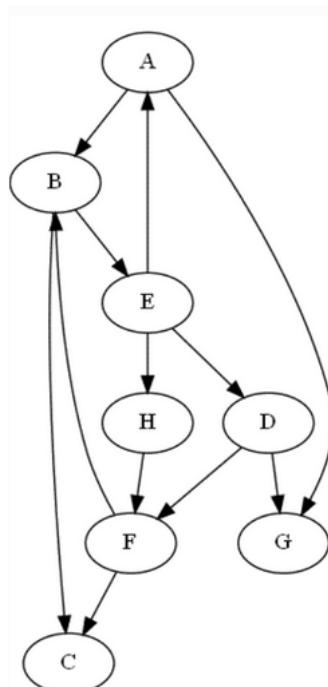
Utiliser l'algorithme du parcours en profondeur d'abord pour ce graphe, et donner l'ordre des sommets visités, en sachant qu'il n'y a pas qu'une seule solution.

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 `graphe_orienté`.

13.1.3 Recherche de chaînes ou de chemins

13.1.4 Recherche de cycles