INSA – MS Année 2019-2020
ROUFFIAC Jean-Eudes

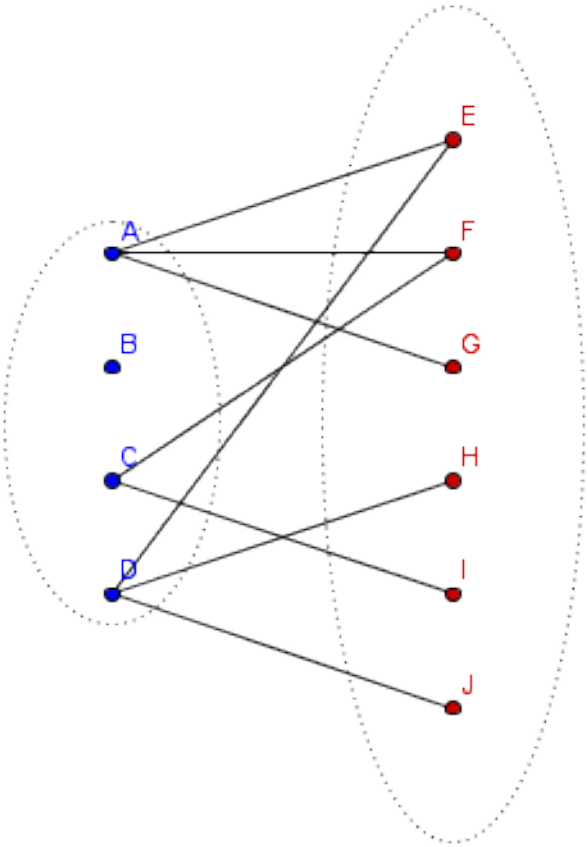
**Algorithmique des graphes**

Rapport de TP n°1

Titre : *Détection des graphes bipartis*

# 1 Le problème

Un graphe est dit biparti si on peut partager son ensemble de sommets en deux parties A et B tels qu'il n'y ait aucune arête entre éléments de A et aucune arête entre éléments de B.




Dans ce TP, l'objectif est de créer puis d'implémenter un algorithme qui détecte si un graphe est biparti ou non. Si il est biparti, il devra renvoyer les deux ensembles de sommets qui composent le graphe biparti. L'algorithme prendra en entrée la liste des successeurs.

# 2 Résolution du problème

Pour résoudre ce problème, nous utiliserons l'algorithme de parcours de graphe en profondeur "Depth First Search" auquel nous apporterons juste quelques modifications. L'idée est de parcourir les sommets voisin d'un sommet $X$ en les catégorisant d'une couleur opposée à la couleur du sommet $X$. On appelle récursivement la fonction de parcours du graphe en profondeur en inversant à chaque fois les deux ensembles dans les paramètres de la fonction pour que les voisins soient dans la couleur opposée au sommet. Dès qu'un sommet est catégorisé dans une couleur, on l'ajoute à la liste des sommets visités. Au début de la fonction de parcours du graphe en profondeur, on crée une condition qui dit que si le voisin du sommet passé en paramètre de la fonction de parcours en profondeur

est déjà catégorisé et dans la même couleur que le sommet, alors le programme s'arrête car dans un graphe biparti, il n'existe pas d'arête entre les sommets du même ensemble. La complexité est de $O(n + 2m)$.

**début**
  **si** *sommet dans bleu* **alors**
    retourner Faux;
  **fin**
  **si** *sommet pas dans est_visite* **alors**
    ajouter le sommet dans est_visite;
    ajouter le sommet dans l'ensemble rouge;
    **pour** *n dans l'ensemble des voisins du sommet* **faire**
      est_biparti(liste_successeur, n, est_visite, bleu, rouge);
    **fin**
  **fin**
  **retourner** *Vrai, rouge, bleu*
**fin**

**Algorithme 1 :** est_biparti(liste_successeur, sommet, est_visite = {}, rouge = {}, bleu = {})

Le code complet en python est le suivant :

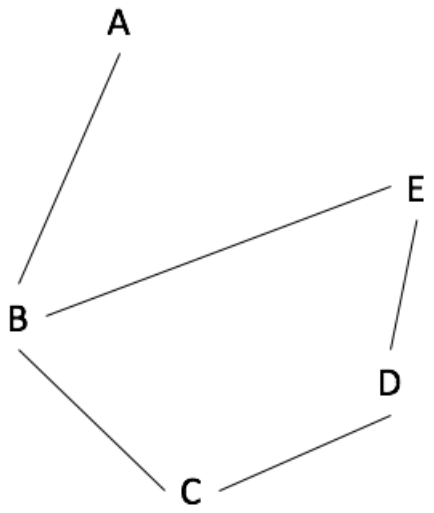
```python
import sys


graph = {
    'A' : ['B'],
    'B' : ['C', 'D', 'A'],
    'C' : ['B', 'A'],
    'D' : ['B', 'A']

}


def est_biparti(list_succ, sommet, est_visite = set() , rouge = set() , bleu = set()):
    if sommet in bleu:
       sys.exit("Pas bi parti")

    if sommet not in est_visite:
        est_visite.add(sommet)
        rouge.add(sommet)
        for n in list_succ[sommet]:
             est_biparti(list_succ, n, est_visite, bleu, rouge)
```

```
22
23      return (True, rouge, bleu)
24
25  result, rouge, bleu = est_biparti(graph,'A')
26
27  if result:
28      print("Le graphe est biparti, ses deux parties sont : ")
29      print(rouge)
30      print(bleu)
```

# 3 Résultats

## 3.1 Exemple 1

Prenons le graphe suivant :




En entrée de l'algorithme, on donne la liste des successeurs. Ainsi, en python, on représente le graphe de la manière suivante :

```
1  graph = {
2      'A' : ['B'],
3      'B' : ['A', 'E', 'C'],
4      'C' : ['B', 'D'],
5      'D' : ['C', 'E'],
```

```
    'E' : ['B', 'D']

}
```

On appelle alors notre fonction avec entrée l'objet graph, ainsi que le 1er élément à savoir le sommet 'A'.

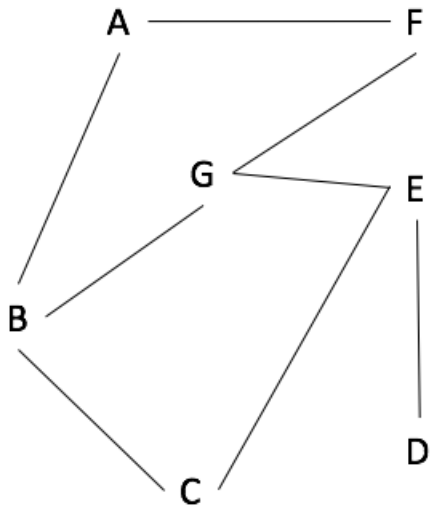
```
result, rouge, bleu = est_biparti(graph,'A')
```

Le résultat donné par le programme est alors le suivant :

```
Le graphe est biparti, ses deux parties sont :
{'C', 'E', 'A'}
{'B', 'D'}
```

## 3.2 Exemple 2

Prenons le graphe suivant :




En entrée de l'algorithme, on donne la liste des successeurs. Ainsi, en python, on représente le graphe de la manière suivante :

```
graph = {
    'A' : ['B', 'F'],
    'B' : ['A', 'C', 'G'],
    'C' : ['B', 'E'],
    'D' : ['E'],
    'E' : ['C', 'D', 'G']
    'F' : ['A', 'G']
    'G' : ['B', 'E', 'F']
```

```
9
10  }
```

On appelle alors notre fonction avec entrée l'objet graph, ainsi que le 1er élément à savoir le sommet 'A'.

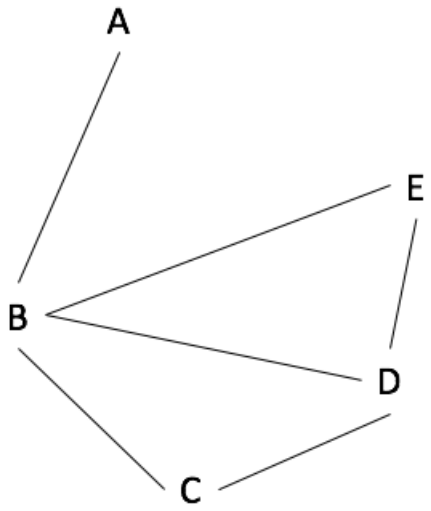
```
1  result, rouge, bleu = est_biparti(graph,'A')
```

Le résultat donné par le programme est alors le suivant :

```
Le graphe est biparti, ses deux parties sont :
{'C', 'G', 'A', 'D'}
{'F', 'B', 'E'}
```

## 3.3 Exemple 3

Reprenons le graphe de l'exemple 1 en ajoutant une arête entre les sommets B et D :




En entrée de l'algorithme, on donne la liste des successeurs. Ainsi, en python, on représente le graphe de la manière suivante :

```
1  graph = {
2      'A' : ['B'],
3      'B' : ['A', 'E', 'C', 'D'],
4      'C' : ['B', 'D'],
5      'D' : ['B', 'C', 'E'],
6      'E' : ['B', 'D']
7
8  }
```

On appelle alors notre fonction avec entrée l'objet graph, ainsi que le 1er élément à savoir le sommet 'A'.

```
result, rouge, bleu = est_biparti(graph,'A')
```

Le résultat donné par le programme est alors le suivant :

```
An exception has occurred, use %tb to see the full traceback.

SystemExit: Pas bi parti
```

L'algorithme s'est donc arrêté car deux sommets voisins se sont vus attribuer la même couleur.