

## Algorithmique (AL5)

### TD n° 2 : parcours en largeur de graphes non orientés

*Les exercices notés avec une étoile (\*) présentent un niveau de difficulté plus élevé.*

#### Exercice 1 : parcours en largeur

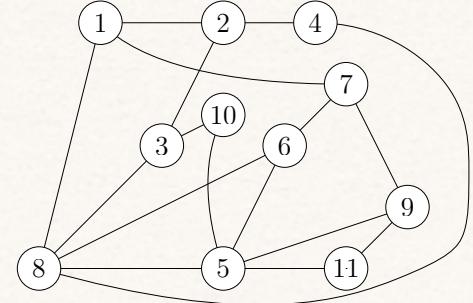
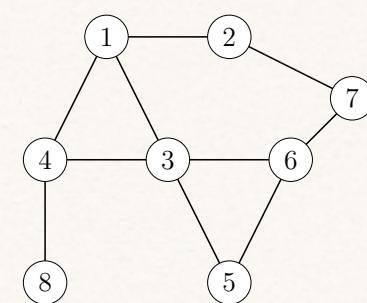
Appliquer aux deux graphes ci-dessous l'algorithme de parcours en largeur à partir du sommet 1. On suppose que le graphe est présenté par liste d'adjacences et que celles-ci sont triées par étiquette croissante de sommets. Pour chaque graphe, donner l'arbre résultant de ce parcours.

##### Parcours en largeur (BFS<sup>1</sup>)

```

Entrées : graphe  $G = (V, E)$  et sommet  $s \in V$ 
début
  créer file( $Q$ ) ;
  marquer( $s$ ) ;
  enfiler( $Q, s$ ) ;
  tant que  $Q \neq \emptyset$  faire
     $u \leftarrow$  défiler( $Q$ ) ;
    pour tous les  $uv \in E$  faire
      si  $v$  non marqué alors
        marquer( $v$ ) ;
        enfiler( $Q, v$ )
  
```

1. Breadth-first search



Algorithme de parcours en largeur

Graphe  $G_1$

Graphe  $G_2$

#### Exercice 2 : parcours et matrice d'adjacence

Quelle est la complexité de l'algorithme de parcours en largeur d'un graphe non orienté si celui-ci est représenté par une matrice d'adjacence ?

#### Exercice 3 : plus courts chemins

Dans cet exercice  $G := (V, E)$  est un graphe quelconque (représenté par une liste d'adjacence), et  $s \in V$ , un de ses sommets. On a  $(|V|, |E|) = (n, m)$ . On définit la *distance de plus court chemin* entre deux noeuds  $v, v' \in V$  comme étant le nombre minimum d'arcs d'une chaîne reliant le sommet  $v$  au sommet  $v'$ , ou  $\infty$  s'il n'existe aucune chaîne de  $v$  à  $v'$ .

Remarque : dans le cas des graphes orientés on ne parle pas de *chaîne*, mais de *chemin*. Par abus de langage, on appelle souvent chemins les chaînes des graphes non orientés.

1. Proposer un algorithme de complexité en  $O(n + m)$  (nombre de sommets plus nombre d'arêtes) qui calcule la distance entre  $s$  et les autres sommets de  $G$ .
2. En proposant une manière astucieuse d'encoder la sortie, en déduire un algorithme de complexité en  $O(n + m)$  qui calcule un chemin le plus court entre  $s$  et chacun des autres sommets de  $G$ .

**Exercice 4 : graphes bipartis**

Un graphe  $G = (V, E)$  est un graphe biparti si l'ensemble des sommets  $V$  peut être partitionné en deux sous-ensembles  $V_1$  et  $V_2$  (i.e.  $V_1 \cap V_2 = \emptyset$  et  $V_1 \cup V_2 = V$ ), de sorte que les arêtes de  $E$  ont exactement une extrémité dans  $V_1$  et une extrémité dans  $V_2$ .

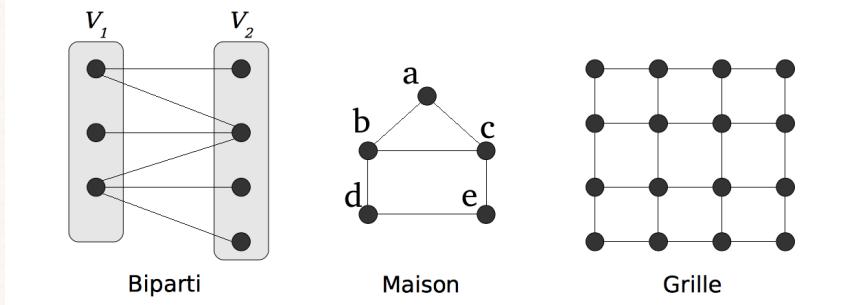


FIGURE 1 – Schéma général d'un graphe biparti, la maison, la grille  $G_{4,4}$

1. Déterminer si le graphe maison est un graphe biparti.
2. Même question pour la grille  $G_{4,4}$  (on pourra nommer les sommets  $(i, j)$  selon leur coordonnées).
3. Montrer qu'un graphe est biparti si et seulement si il n'a pas de cycle de longueur impaire. La longueur d'un cycle est le nombre d'arêtes qui le composent. (Astuce : on peut se servir d'une 2-coloration).  
*Remarque : colorier un graphe revient à mettre des couleurs sur chaque sommet du graphe. La seule contrainte imposée est que deux sommets adjacents ne soient pas de la même couleur.*
4. Proposer un algorithme qui prend en entrée un graphe, qui teste si le graphe est biparti, et qui renvoie un cycle impair sinon. On pourra se servir du parcours en largeur (BFS) vu en cours.
5. Évaluer la complexité de votre algorithme.

**Exercice 5 : cycle (\*)**

Proposer un algorithme renvoyant, s'il existe, un cycle de longueur minimale passant par un sommet  $s$  donné dans un graphe  $G$  non orienté. Prouver sa correction. On pourra dans un premier temps supposer que le graphe est biparti.

# Ex 1 : How to CM2 pg 40

## Exercice 1 : parcours en largeur

Appliquer aux deux graphes ci-dessous l'algorithme de parcours en largeur à partir du sommet 1. On suppose que le graphe est présenté par liste d'adjacences et que celles-ci sont triées par étiquette croissante de sommets. Pour chaque graphe, donner l'arbre résultant de ce parcours.

### Parcours en largeur (BFS<sup>1</sup>)

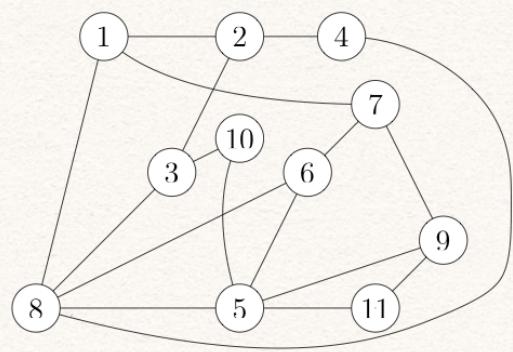
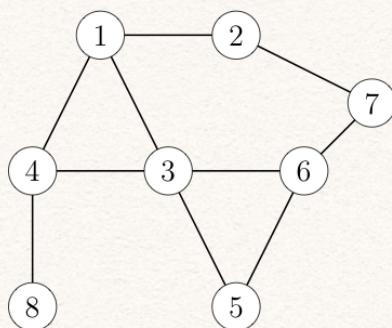
**Entrées :** graphe  $G = (V, E)$  et sommet  $s \in V$

**début**

```

    créer file( $Q$ ) ;
    marquer( $s$ ) ;
    enfiler( $Q, s$ ) ;
    tant que  $Q \neq \emptyset$  faire
         $u \leftarrow$  dépiler( $Q$ ) ;
        pour tous les  $uv \in E$  faire
            si  $v$  non marqué alors
                marquer( $v$ ) ;
                enfiler( $Q, v$ )
    
```

1. Breadth-first search

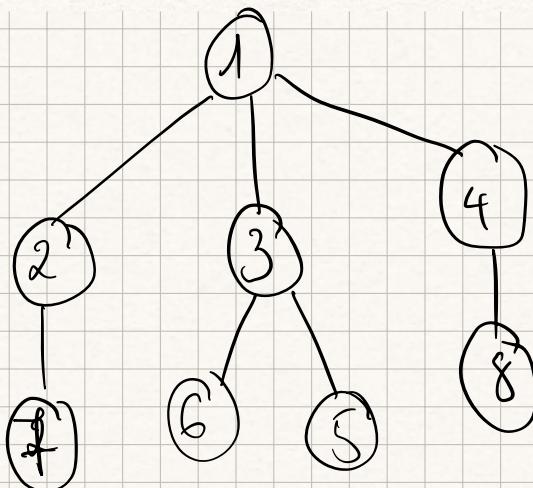


Algorithme de parcours en largeur

Graphe  $G_1$

Graphe  $G_2$

Graphique 1



$Q = []$

marq 1

$u = 1, Q = [ ]$

marq 2,

$Q = [2, 3, 4]$

$Q = [3, 4]$

$u = 2$

marq ?

$Q = [3, 4, ?]$

défile 3

$Q = [4, ?]$

$u = 3$

marq 5 et 6

$Q = [4, ?, 5, 6]$

défile 4

$Q = [?, 5, 6]$

$u = 4$

marq 8

$Q = [7, 5, 6, 8]$

défile ?

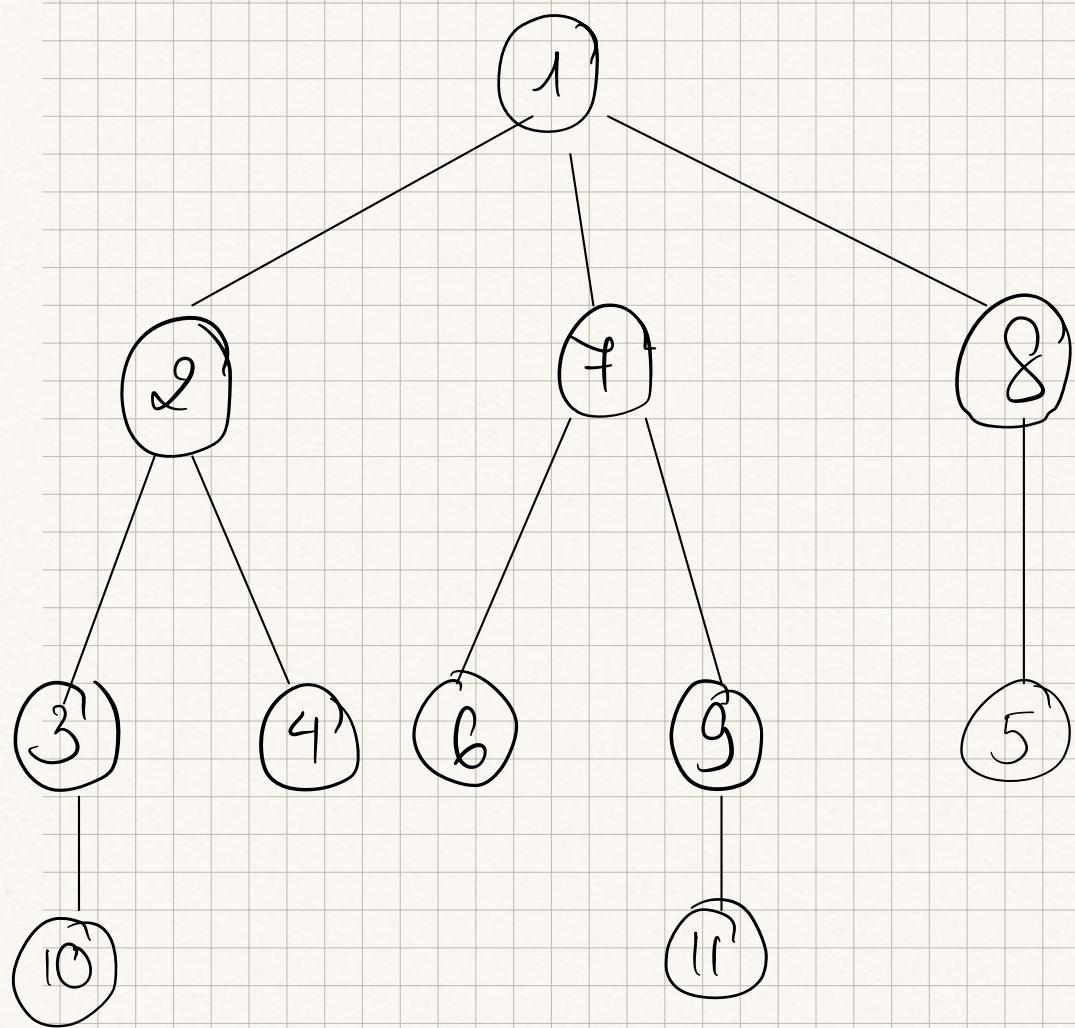
$Q = [5, 6, 8]$

marq rien

défile 5

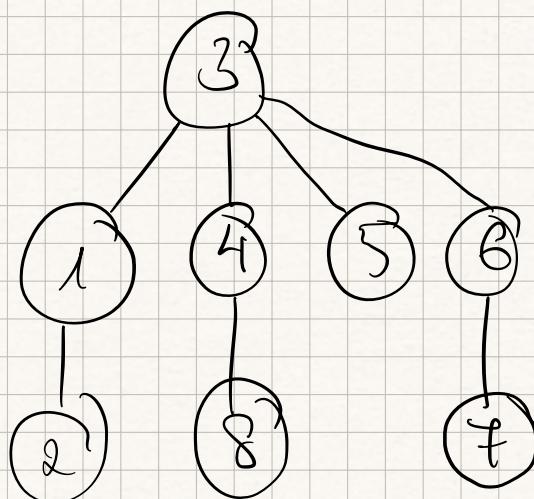
$Q = [6, 8]$

## Graphique 2



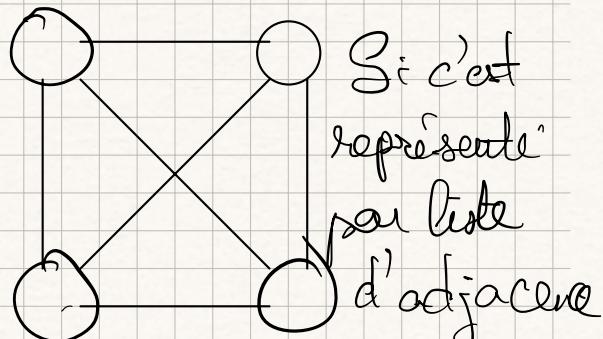
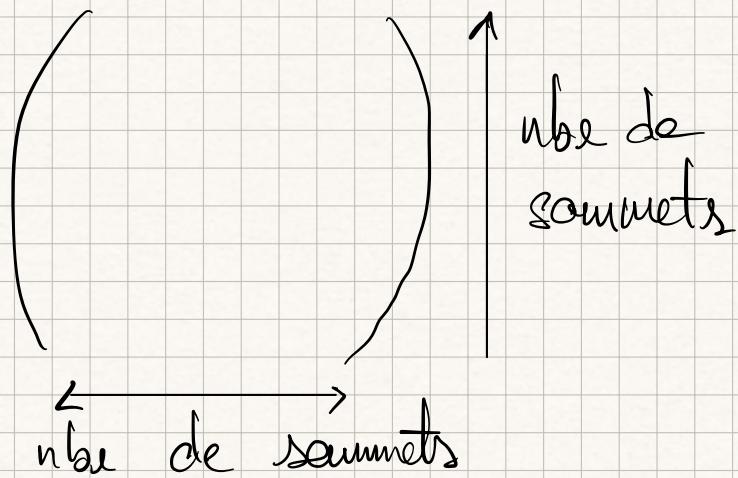
mais rien  
défile 6  
 $Q = [8]$   
défile 8  
mais rien  
 $Q = [ ]$

## Graphique 1 : Autre solution



Ex2

$G$  représente matrice d'adjacence



$$\mathcal{O}(|E| + |V|)$$

Complexité du parcours en largeur ?

$$\mathcal{O}(|E|^2)$$

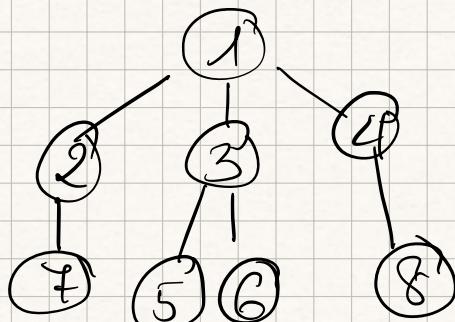
$\mathcal{O}(|V| + |E|)$   
l'ensemble des sommets  
n<sup>o</sup> d'arête

Ex3

$$G = (V, E)$$

dist.  $(v_i, v_j) \stackrel{\text{nb}}{=} \text{minimum d'arc entre } v_i \text{ et } v_j$   
(il n'en existe pas)

Ex1 graphe 1



Sommets	2	3	4	5	6	7	8
---------	---	---	---	---	---	---	---

dist min	1	1	1	2	2	2	2
/1							

$$\pi_v \leftarrow []$$

$$[+\infty, \dots +\infty]$$

Parcours en largeur

À chaque fois qu'on marque un sommet  $\omega$  venant d'un père  $u$

$$d(\omega) \leftarrow d(u) + 1$$

Un graphe  $G = (V, E)$  est un graphe biparti si l'ensemble des sommets  $V$  peut être partitionné en deux sous-ensembles  $V_1$  et  $V_2$  (i.e.  $V_1 \cap V_2 = \emptyset$  et  $V_1 \cup V_2 = V$ ), de sorte que les arêtes de  $E$  ont exactement une extrémité dans  $V_1$  et une extrémité dans  $V_2$ .

Ex 4

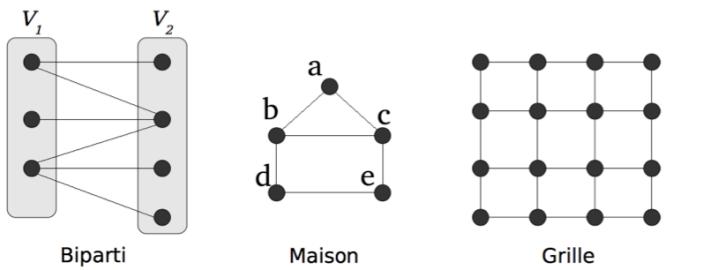


FIGURE 1 – Schéma général d'un graphe biparti, la maison, la grille  $G_{4,4}$

1. Déterminer si le graphe maison est un graphe biparti.
2. Même question pour la grille  $G_{4,4}$  (on pourra nommer les sommets  $(i, j)$  selon leur coordonnées).
3. Montrer qu'un graphe est biparti si et seulement si il n'a pas de cycle de longueur impaire. La longueur d'un cycle est le nombre d'arêtes qui le composent. (Astuce : on peut se servir d'une 2-coloration).

Remarque : colorier un graphe revient à mettre des couleurs sur chaque sommet du graphe. La seule contrainte imposée est que deux sommets adjacents ne soient pas de la même couleur.

4. Proposer un algorithme qui prend en entrée un graphe, qui teste si le graphe est biparti, et qui renvoie un cycle impair sinon. On pourra se servir du parcours en largeur (BFS) vu en cours.
5. Évaluer la complexité de votre algorithme.

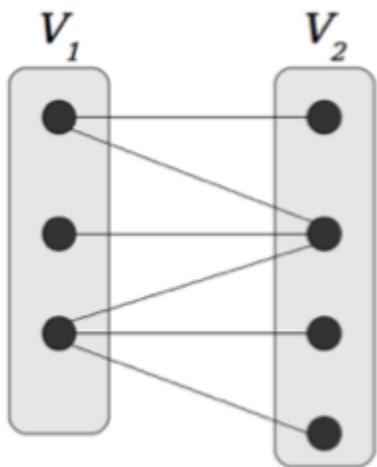
1, Non

Un graphe biparti est un type de graphe où les sommets peuvent être divisés en deux ensembles distincts, de sorte qu'il n'y ait pas de bords (ou liens) entre les sommets du même ensemble.

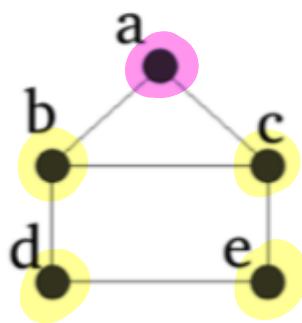
En termes simples : Imaginez deux groupes de points. Les points d'un groupe peuvent être connectés aux points de l'autre groupe, mais jamais entre eux au sein du même groupe.

2<sup>y</sup> colonne  $\text{col} = V \rightarrow \{R, V\}$

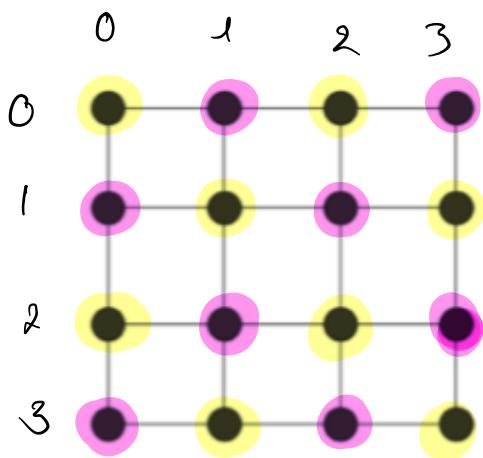
combinaisons  
 $\begin{cases} \text{col}(a) \neq \text{col}(b) \\ \text{col}(b) \neq \text{col}(c) \\ \text{col}(c) \neq \text{col}(a) \end{cases}$  alors



Biparti



Maison



Grille

$$\text{col}(i, j) = \begin{cases} R & \text{ssi } i \equiv 1 \pmod 2 \\ V & \text{sinon} \end{cases}$$

3<sup>y</sup>

$\Rightarrow$  Soit  $G$  un graphe biparti

Supposons qu'il existe un cycle  $C$

Montrons que  $|C|$  est paire

$$C = (v_1, e_1, v_2, \dots, v_n, e_n, v_1) \rightarrow \text{cycle}$$

Supposons  $n$  impair

Soit  $\text{cal}: V \rightarrow \{R, V\}$  de  $G$ :  $\underbrace{\text{cal}(v_i) = \text{cal}(v_j)}$

$$\Leftrightarrow i \equiv j \pmod{2}$$

$$\text{cal}(v_n) = \text{cal}(v_1)$$

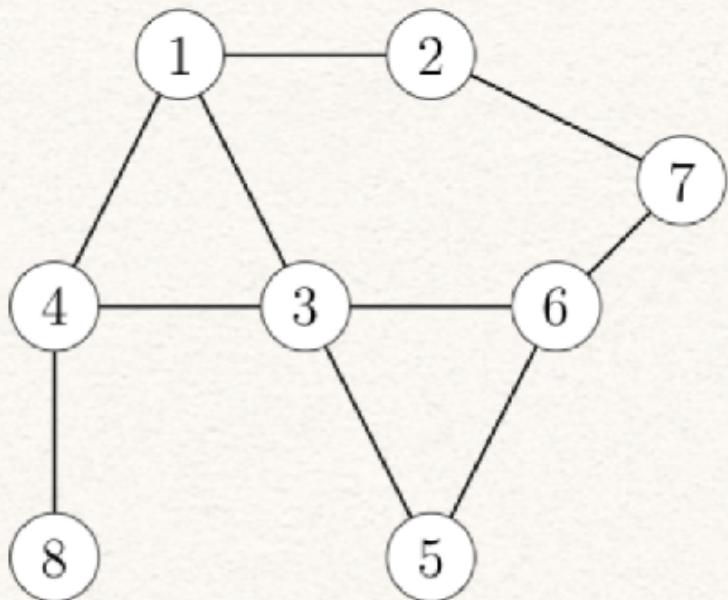
$$\text{cal}(v_n) \neq \text{cal}(v_1)$$

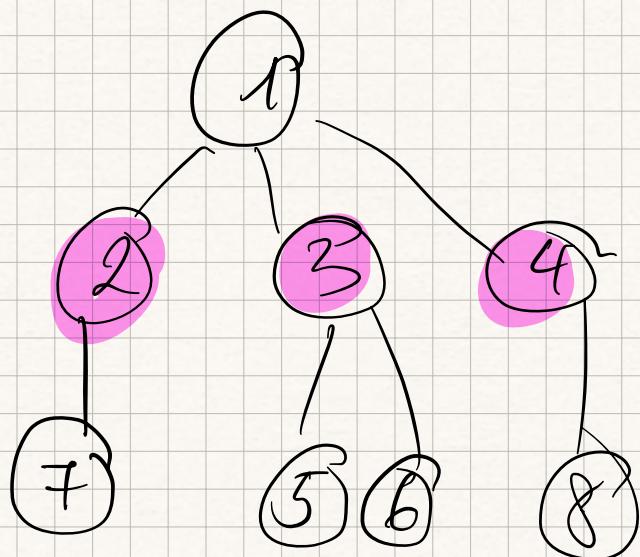
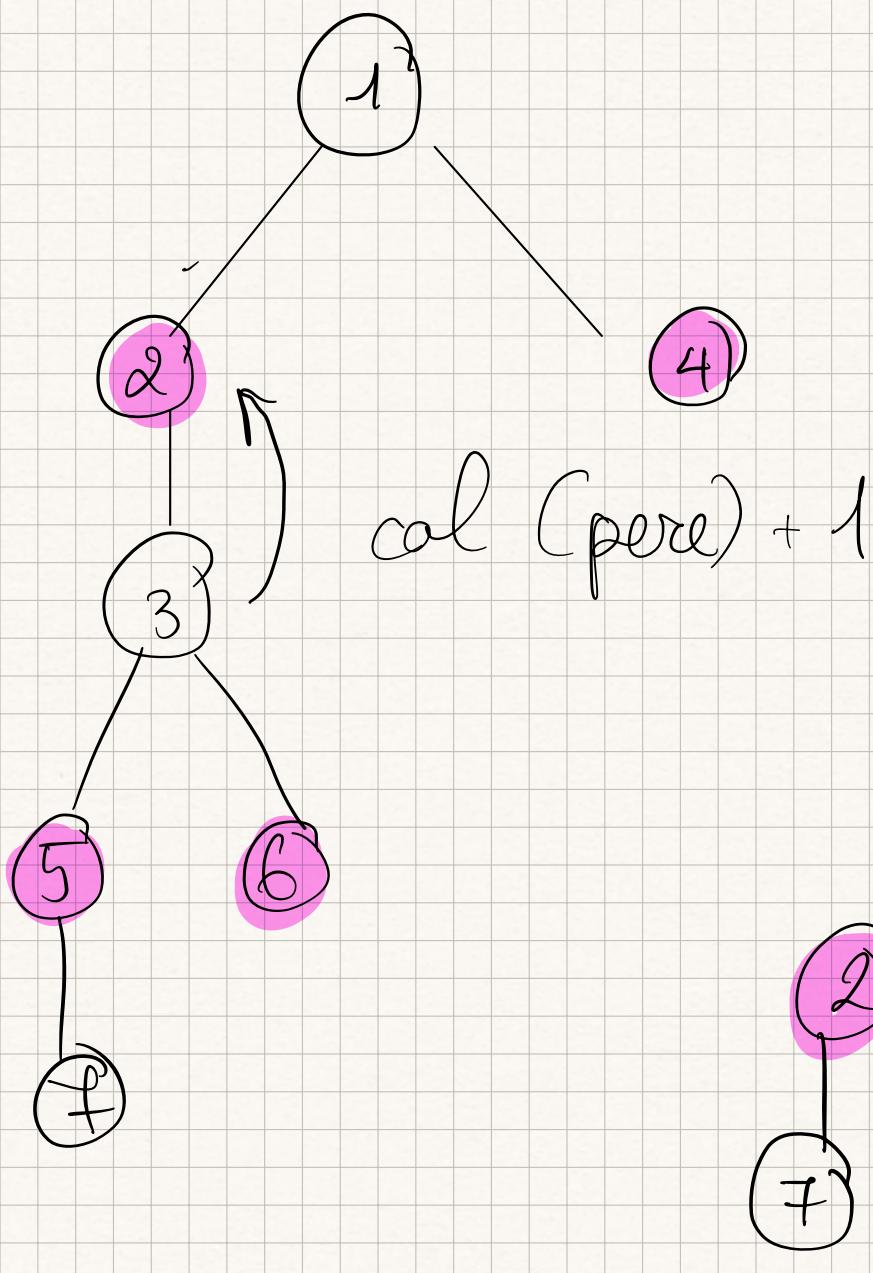
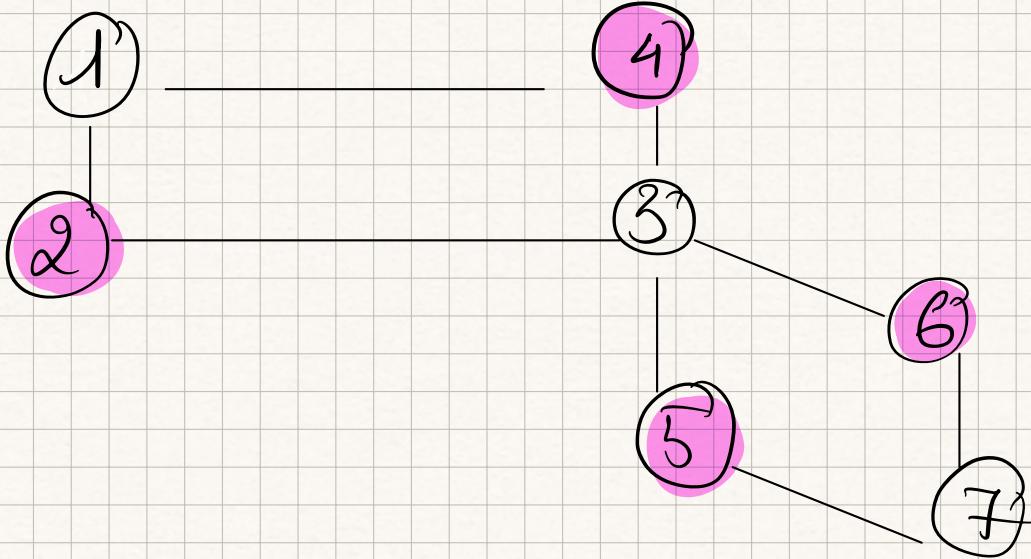
absurde

$\Leftarrow$  Soit  $G$  un graphe sans cycle de longueur impaire  
(supp  $G$  connexe)

{ Parcours en largeur 2 cal

$$\text{cal}(i, j) = R \text{ssi } i \equiv j \pmod{2}$$





$E : G = (V, E)$ , sommet  $S \in V$   
creer file  $Q$ ;

$\text{col}(S) = 0$ ;

enfiler  $(Q, S)$ ;

Tant que  $Q \neq \emptyset$ ;

$u \leftarrow \text{defiles}(Q)$ ;

Pour tout  $(u, v)$  dans  $E$ :

Si  $\text{col}(v)$  non défini:

$$\text{col}(v) = \text{col}(u) + 1 \bmod 2;$$

enfiler  $(Q, v)$

Soit  $v_1, v_2 \in E$  avec  $\text{col}(v_1) = \text{col}(v_2)$

Il existe chemin  $C_1$  de  $v$  à  $v_1$ ,  $C_2$  de  $v$  à  $v_2$  de  
même parité

$$C_1, [v_1], \overbrace{\{v_1, v_2\}}^{E}, \overbrace{v_2}^{E}, C_2$$

impair

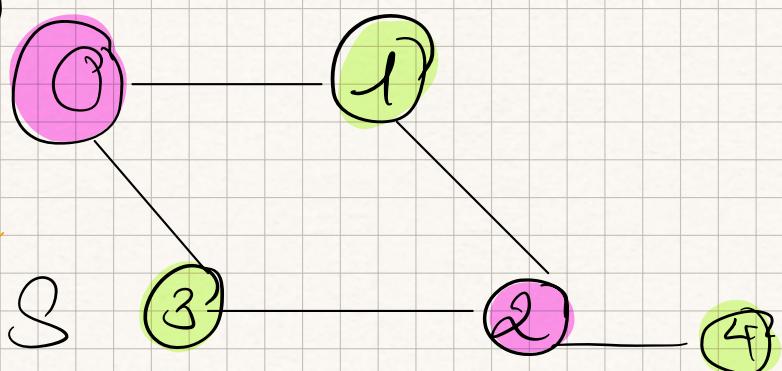
Ex 5 :

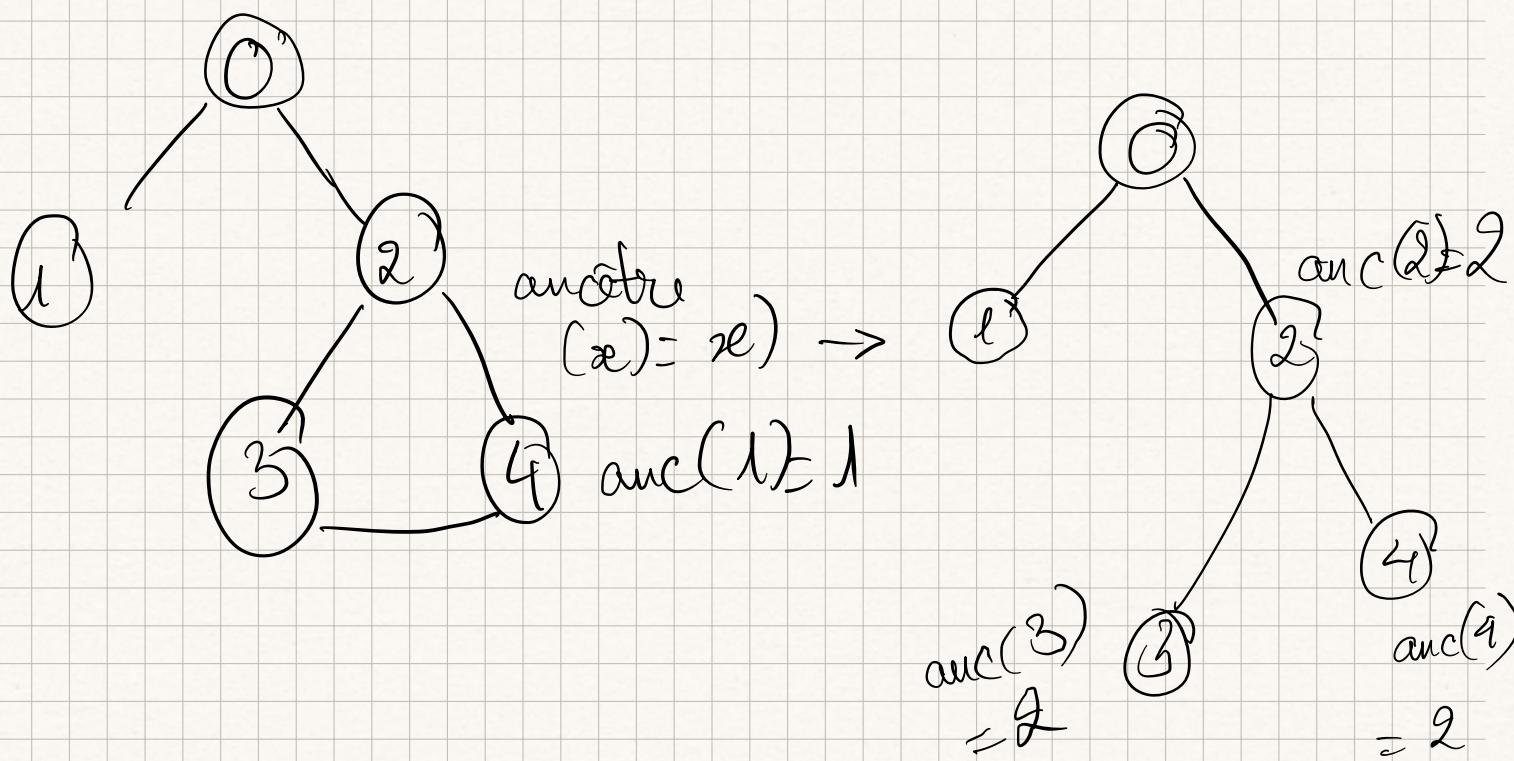
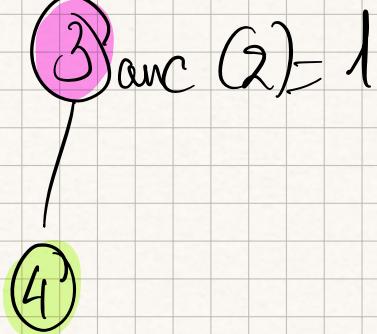
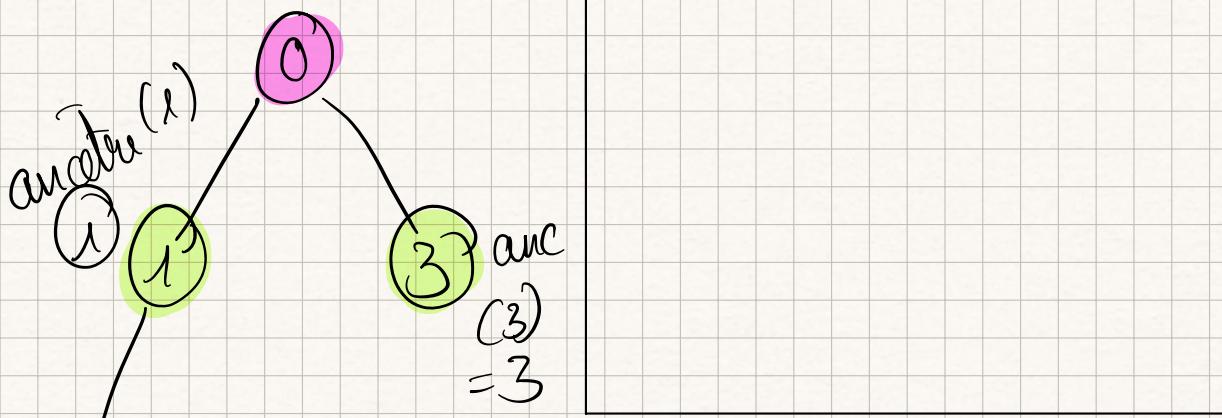
Algo :

Entrée : graphe  $G$  orienté un sommet  $S$

(Supp  $G$  bipartite)

Sortie = un cycle de longueur minimale si il existe passant par  $S$





File = vide

Pour tout  $u$  dans  $V$

couleur ( $u$ ) = blanc  
pere ( $u$ ) = vide

ancêtre ( $u$ ) = vide

push ( $S, Q$ )  
 $lq = +\infty$

couple = vide  
Tant que  $Q \neq \emptyset$   
 $x = \text{Pop}(Q)$   
Si  $\text{père}(x) = S$   
ancêtre( $x$ ) =  $x$

Pour tout voisin  $y$  de  $x$   
o Si  $\text{col}(y) = \boxed{b}$   
 $\text{cal}(y) = g$   
 $\text{père}(y) = x$   
 $\text{dist}(y) = \text{dist}(x) + 1$   
 $\text{anc}(y) = \text{anc}(x)$   
 $\text{push}(y, Q)$

Si  $\text{col}(y) = g$   
Si  $\text{anc}(x) \neq \text{anc}(y)$   
et  $\text{dist}(x) + \text{dist}(y) + 1 < lg$   
 $lg = \text{dist}(x) + \text{dist}(y) + 1$   
 $\text{couple}(x, y)$   
 $\text{cal}(x) = y$  nœu

$x, y = \text{couple}$   
 $\text{return}(x, y) + \text{les 2 chemins de } x \text{ et } y$   
vers