Algorithmique et Complexité
TD 1/7 – Parcours de graphes

CentraleSupélec – Gif

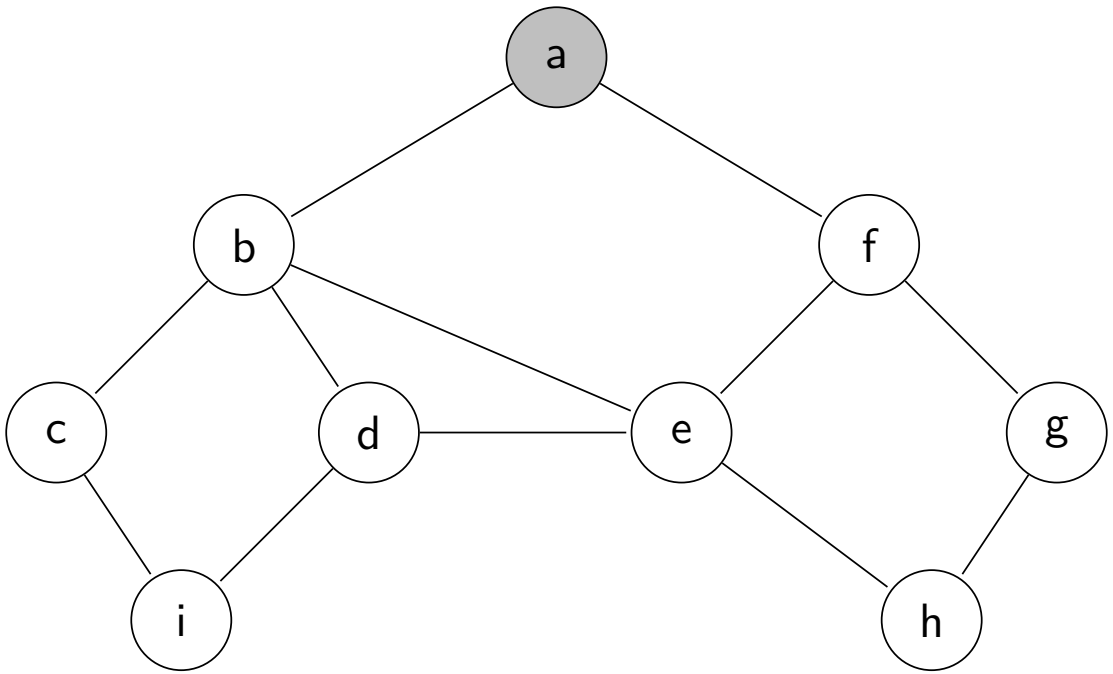
ST2 – Gif

Plan

Question 1

Question : En vous appuyant sur les algorithmes de parcours de graphe vus en cours, proposez un algorithme qui calcule la distance en nombre d'arêtes entre un sommet du graphe (désigné comme racine) et tous les autres sommets (on parle de *profondeur*).

Question 1 : un graphe



Question 1 : Exploration en largeur

```
def BFS(V,E,s,t):
    lnext = [s]                    # la file
    reached = { s : True }
    while len(lnext)>0:
      n = pop_begin(lnext)
      if n==t:
        return True
      for v in neighbours(n,E):
        if not v in reached:
          reached[v] = True
          add_end(v,lnext)
    return False
```

Question 1 : correction

```
def BFS_depth(V,E,s):
    depth = {} # dispense de visited
    depth[s] = 0
    next = [s]
    while len(next)>0:
      n = pop_begin(next)
      for v in neighbours(n,E):
        if not v in depth:
          add_end(v,next)
          depth[v] = depth[n] + 1
    return depth
```

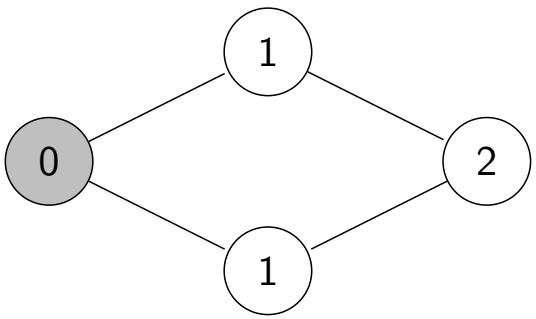
Question 1 : correction

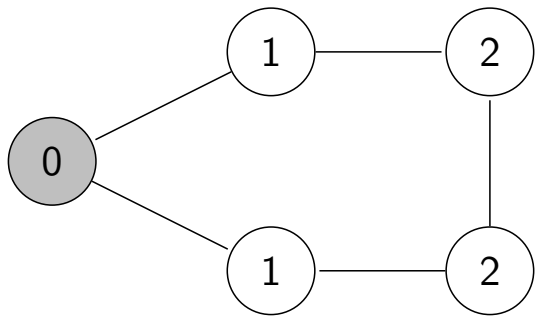
On peut remarquer que `depth` dispense et encode la liste des nœuds visités/atteints et permet ainsi d'éviter les cycles et de ne pas ajouter plusieurs fois des voisins qui ont plusieurs parents dans le parcours. C'est toujours ça de pris ! Si G est représenté par liste d'adjacence et suffisemment dense, alors la complexité est $\mathcal{O}(|V| + |E|) \approx \mathcal{O}(|E|)$. Si G est représenté par matrice d'adjacence, alors la complexité est $\mathcal{O}(|V|^2)$.

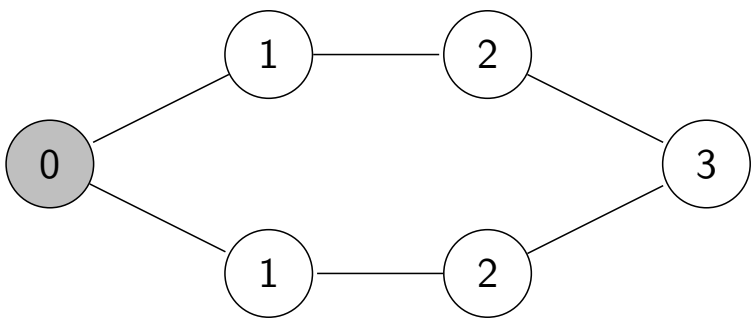
Attention : l'algorithme proposé ci-dessus ne fonctionne que pour un graphe connexe. Si le graphe n'est pas connexe, il faut traiter chaque composante séparément, comme vu en cours dans l'algorithme d'énumération des composantes connexes.

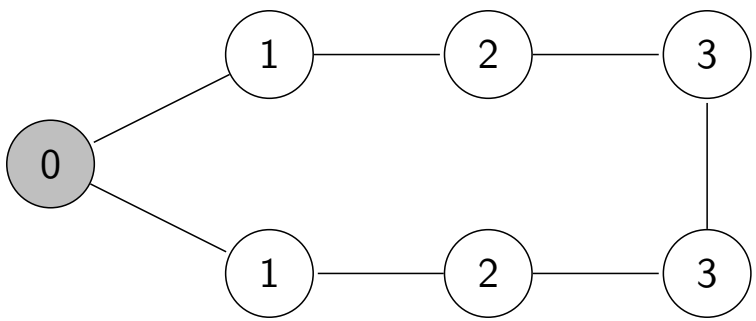
Question 2

Considérons les cycles suivants de longueur paires et impaires pour lesquels nous avons calculé les profondeurs :









Question 2

Question :

- Que remarquez vous (sur les profondeurs) dans les cas des graphes avec un cycle **impair** (en nombre d'arêtes) ?
- Est-ce une propriété caractéristique ?
- Énoncez le cas général et donnez une preuve !

Question 2 : correction

Propriété caractéristique : Un graphe contient un cycle C avec un nombre impair d'arêtes ssi :

$$\exists (x, y) \in E. \quad depth(x) = depth(y)$$

Question 2 : correction

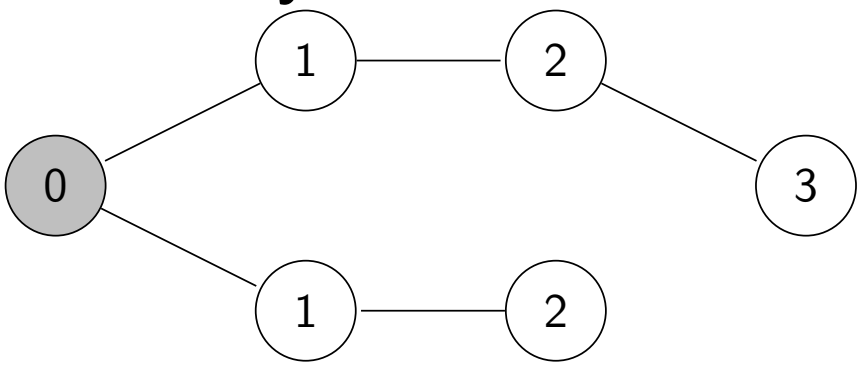
Remarque :
Dans un graphe quelconque, toutes les arêtes relient des sommets de profondeurs "voisines" :

$$\forall (x, y) \in E. \qquad |depth(x) - depth(y)| \leq 1$$

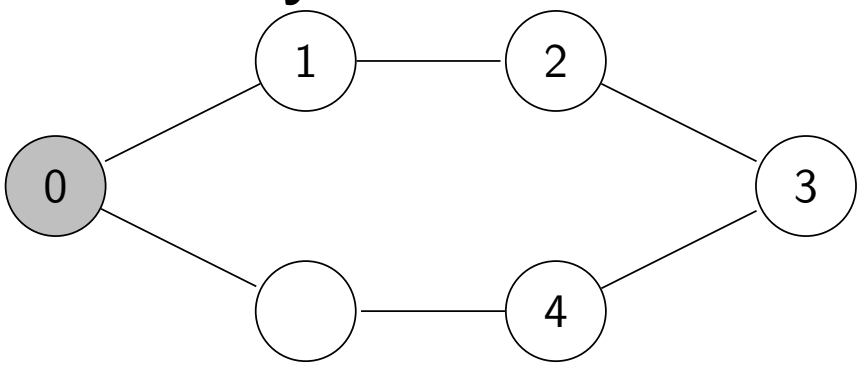
Question 2 : correction

Pourquoi plus ou moins 1 (pourquoi pas 2 ou 3 ou plus) ?

1. **Cas sans cycle** :

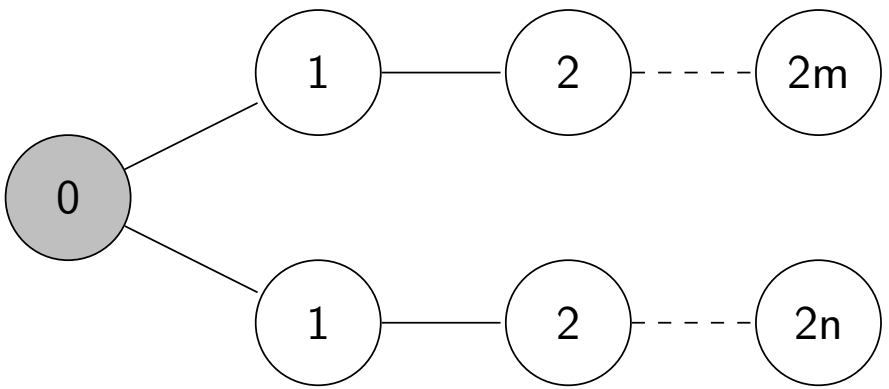


2. **Cas avec cycle** :



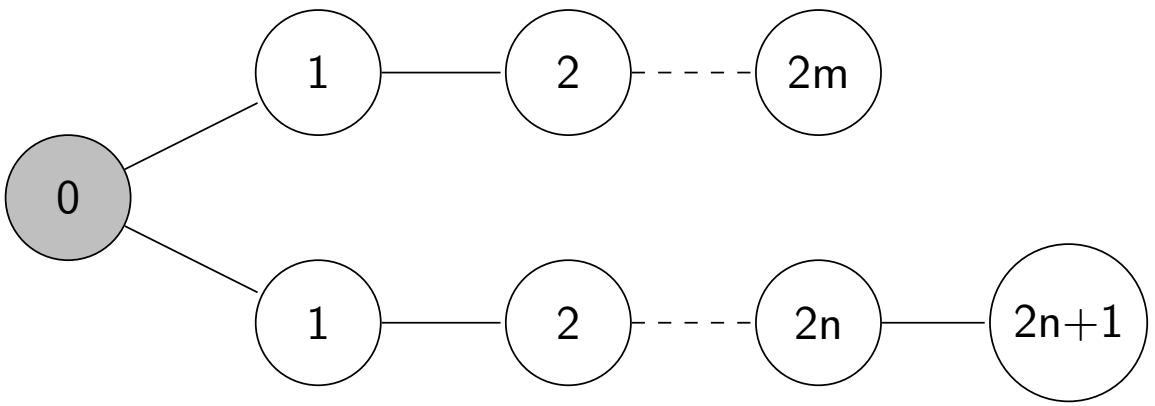
Question 2 : correction

$\Rightarrow$ par contraposée,
on suppose que $\forall (x, y) \in C.\ depth(x) \neq depth(y)$,
alors $\forall (x, y) \in C.\ depth(x) = depth(y) \pm 1$, on aura le long du cycle un nœud de profondeur paire, suivi d'un nœud de profondeur impaire et on alternera ainsi de suite et on ne pourra pas fermer un cycle de taille impaire !



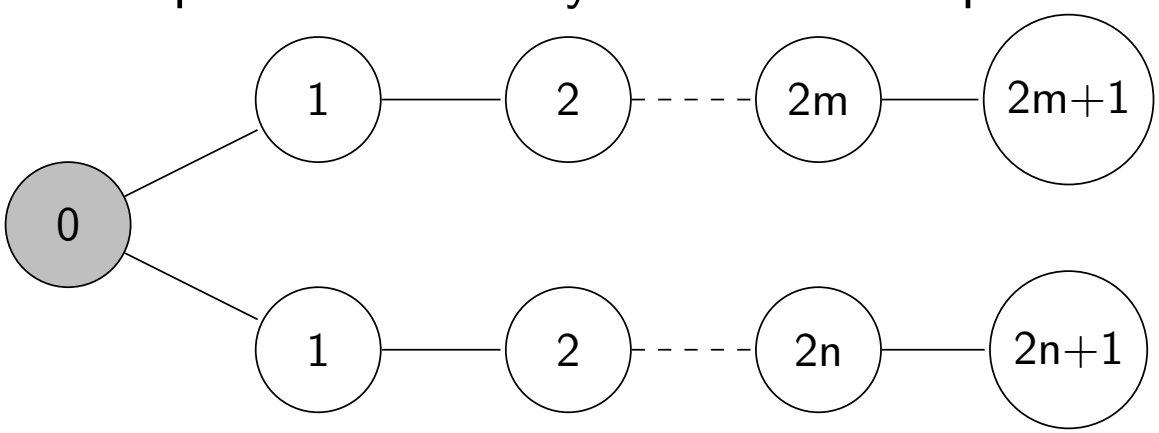
Question 2 : correction

$\Rightarrow$ par contraposée,
on suppose que $\forall (x, y) \in C.\ depth(x) \neq depth(y)$,
alors $\forall (x, y) \in C.\ depth(x) = depth(y) \pm 1$, on aura le long du cycle un nœud de profondeur paire, suivi d'un nœud de profondeur impaire et on alternera ainsi de suite et on ne pourra pas fermer un cycle de taille impaire !



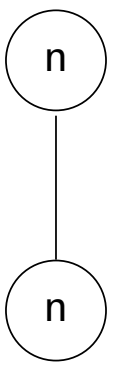
Question 2 : correction

$\Rightarrow$ par contraposée,
on suppose que $\forall (x, y) \in C.\ depth(x) \neq depth(y)$,
alors $\forall (x, y) \in C.\ depth(x) = depth(y) \pm 1$, on aura le long du cycle un nœud de profondeur paire, suivi d'un nœud de profondeur impaire et on alternera ainsi de suite et on ne pourra pas fermer un cycle de taille impaire !



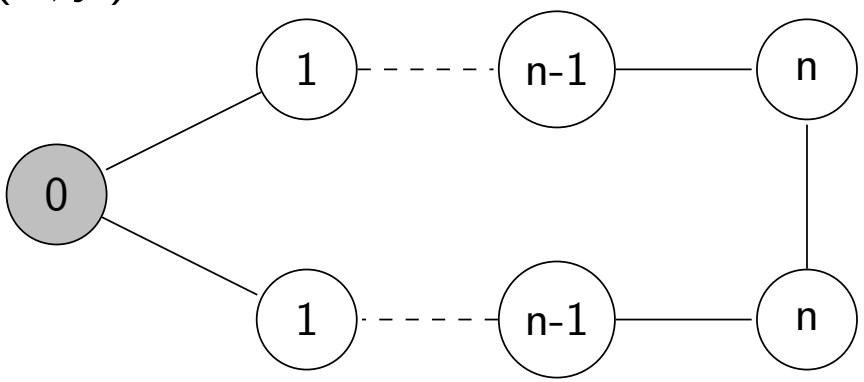
Question 2 : correction

$\Leftarrow$ si il existe une arête $(x, y) \in E$ pour laquelle $depth(x) = depth(y)$.
On considère l'arbre de parcours qui a permis d'annoter les profondeurs.
Dans cet arbre x et y ont un premier ancêtre z en commun (éventuellement la racine) à partir duquel on peut former un cycle impair de taille $(2 \times depth(x)) + 1$ en ajoutant l'arête (x, y) au sous-arbre de z.



Question 2 : correction

$\Leftarrow$ si il existe une arête $(x, y) \in E$ pour laquelle $depth(x) = depth(y)$.
On considère l'arbre de parcours qui a permis d'annoter les profondeurs.
Dans cet arbre x et y ont un premier ancêtre z en commun (éventuellement la racine) à partir duquel on peut former un cycle impair de taille $(2 \times depth(x)) + 1$ en ajoutant l'arête (x, y) au sous-arbre de z.



Question 3

Question : Proposez un algorithme qui détermine si un graphe contient un cycle impair.

Question 3 : correction

On lance l'algo d'étiquetage des profondeurs (question 1) puis pour chaque arête du graphe $(x, y) \in E$, il faut tester si x et y sont de même profondeur :

```
def has_odd_cycle(V,E): # V cannot be empty
    d = BFS_depth(V,E,V[0])
    for (e0, e1) in E:
        if d[e0] == d[e1]:
            return True
    return False
```

Question 4

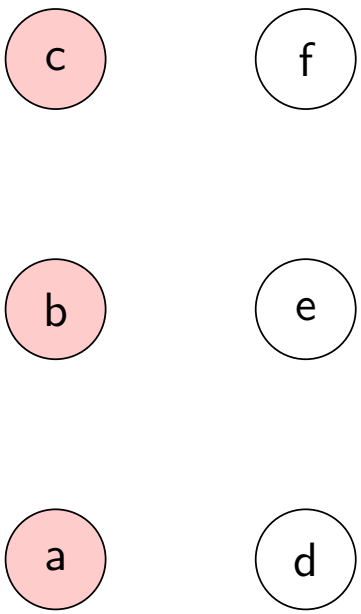
Un graphe $G = (V, E)$ est dit biparti s'il existe une partition de son ensemble de sommets en deux sous-ensembles $V_1 \subseteq V$ et $V_2 \subseteq V$ telle que chaque arête de E a une extrémité dans V_1 et l'autre dans V_2.

Question : Dans un graphe biparti, peut-t-il exister un cycle avec un nombre impair d'arêtes ?

est ce une propriété caractéristique ? Justifiez votre réponse.

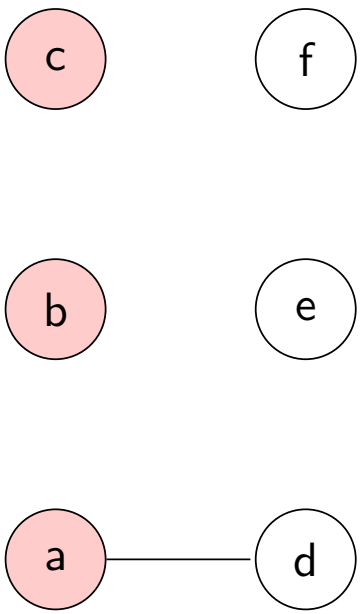
Question 4 : correction

$\Rightarrow$ Si le graphe est biparti, tout chemin alterne entre un sommet de chaque partition donc pour former un cycle et revenir au sommet de départ, il faut nécessairement un nombre pair d'arêtes.



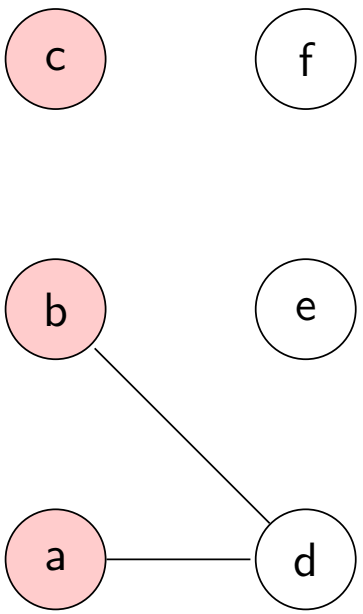
Question 4 : correction

$\Rightarrow$ Si le graphe est biparti, tout chemin alterne entre un sommet de chaque partition donc pour former un cycle et revenir au sommet de départ, il faut nécessairement un nombre pair d'arêtes.



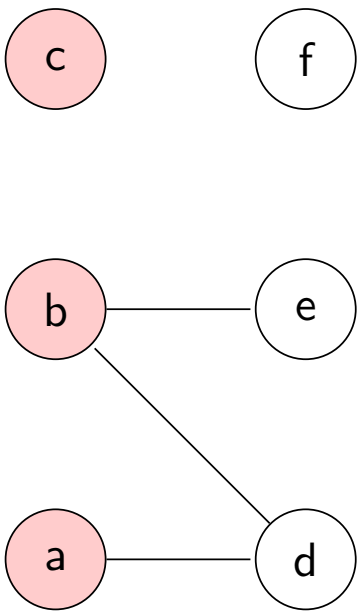
Question 4 : correction

$\Rightarrow$ Si le graphe est biparti, tout chemin alterne entre un sommet de chaque partition donc pour former un cycle et revenir au sommet de départ, il faut nécessairement un nombre pair d'arêtes.



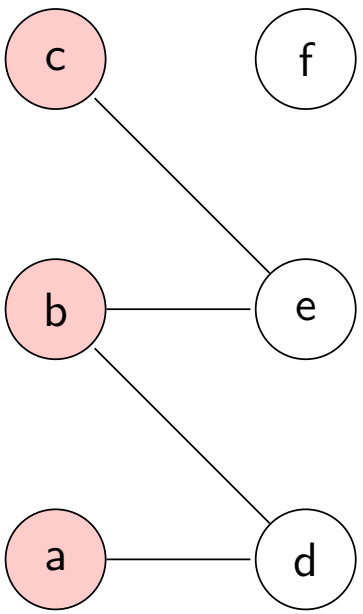
Question 4 : correction

$\Rightarrow$ Si le graphe est biparti, tout chemin alterne entre un sommet de chaque partition donc pour former un cycle et revenir au sommet de départ, il faut nécessairement un nombre pair d'arêtes.



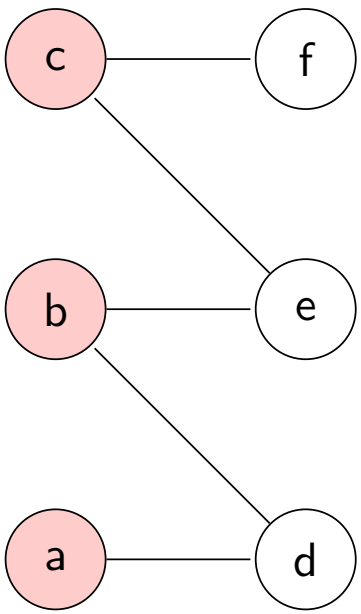
Question 4 : correction

$\Rightarrow$ Si le graphe est biparti, tout chemin alterne entre un sommet de chaque partition donc pour former un cycle et revenir au sommet de départ, il faut nécessairement un nombre pair d'arêtes.



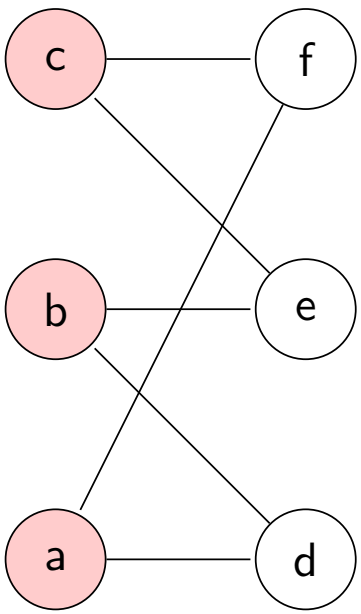
Question 4 : correction

$\Rightarrow$ Si le graphe est biparti, tout chemin alterne entre un sommet de chaque partition donc pour former un cycle et revenir au sommet de départ, il faut nécessairement un nombre pair d'arêtes.



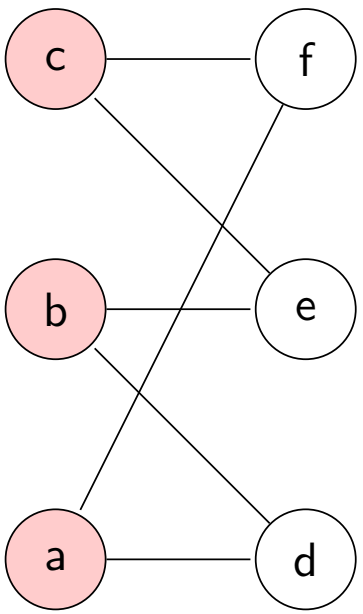
Question 4 : correction

$\Rightarrow$ Si le graphe est biparti, tout chemin alterne entre un sommet de chaque partition donc pour former un cycle et revenir au sommet de départ, il faut nécessairement un nombre pair d'arêtes.



Question 4 : correction

$\Rightarrow$ Si le graphe est biparti, tout chemin alterne entre un sommet de chaque partition donc pour former un cycle et revenir au sommet de départ, il faut nécessairement un nombre pair d'arêtes.



Dans un graphe biparti, tous les cycles sont donc pairs.

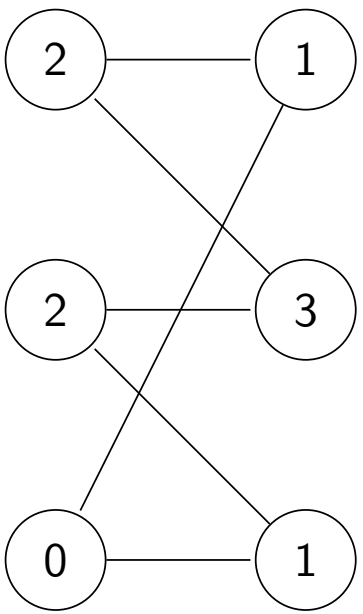
Question 4 : correction

$\Leftarrow$ Supposons qu'il n'existe pas de cycle impair alors (quest. 2) :
Pas de cycle impair (que des cycles pairs) $\Longrightarrow$
$\forall (x, y) \in E. \qquad depth(x) = depth(y) \pm 1$

Question 4 : correction

$\Leftarrow$ Supposons qu'il n'existe pas de cycle impair alors (quest. 2) :
Pas de cycle impair (que des cycles pairs) $\Longrightarrow$
$\forall (x, y) \in E. \qquad depth(x) = depth(y) \pm 1$

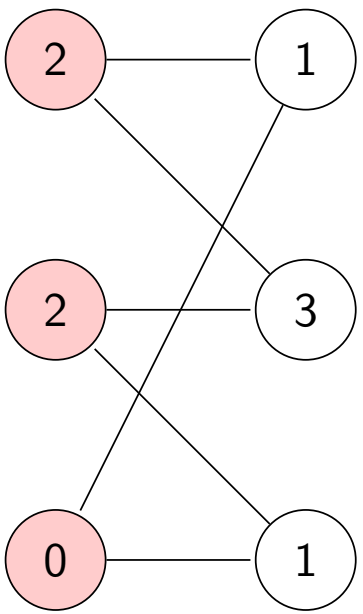
Nous allons considérer la partition des sommets de profondeur paire V_1 d'un côté et impaire V_2 de l'autre.



Question 4 : correction

$\Leftarrow$ Supposons qu'il n'existe pas de cycle impair alors (quest. 2) :
Pas de cycle impair (que des cycles pairs) $\Longrightarrow$
$\forall (x, y) \in E. \qquad depth(x) = depth(y) \pm 1$

Nous allons considérer la partition des sommets de profondeur paire V_1 d'un côté et impaire V_2 de l'autre.

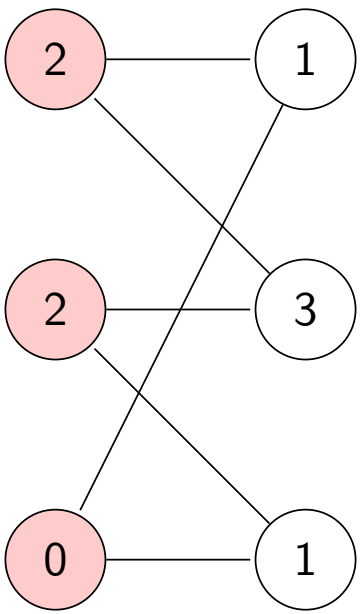


Question 4 : correction

Comme il n'y a aucune arête entre deux sommet de même profondeur ni entre deux sommets de profondeurs différentes de plus de 1 :

$\Longrightarrow$ Alors toutes les arêtes sont dans $V_1 \times V_2$.

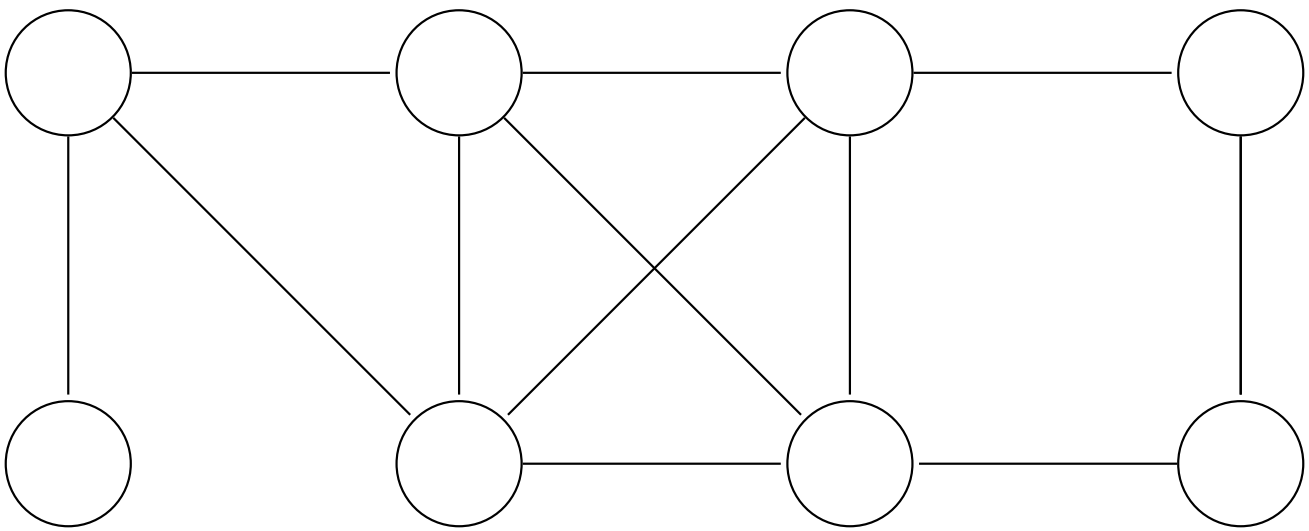
$\Longrightarrow$ Le graphe est donc bien bi-parti.



Question 5

Question : Déduisez un algorithme qui permet de déterminer si un graphe est biparti.

Testez votre algorithme sur le graphe suivant. Est-il biparti ? Justifiez votre réponse

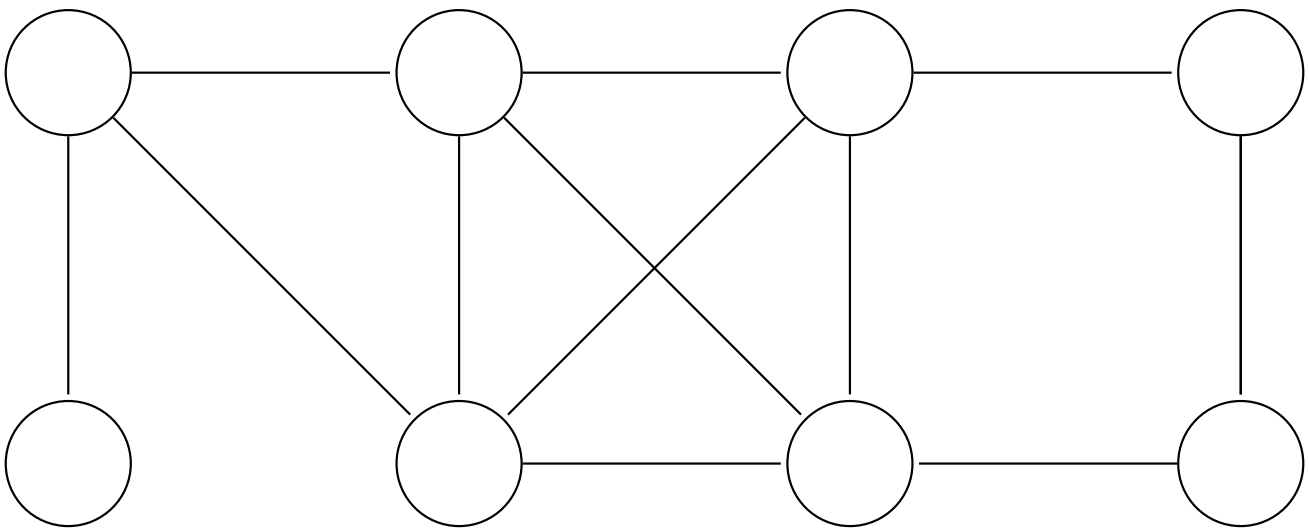


Question 5 : correction

Dans les questions précédentes, nous avons montré :

Bi-parti $\Leftrightarrow_{question4}$ Aucun cycle impair $\Leftrightarrow_{question2}$ aucune paire de sommets voisins de même profondeur

On choisit un nœud au hasard (ici en haut à gauche) et on réalise un parcours en largeur, en annotant les profondeurs,

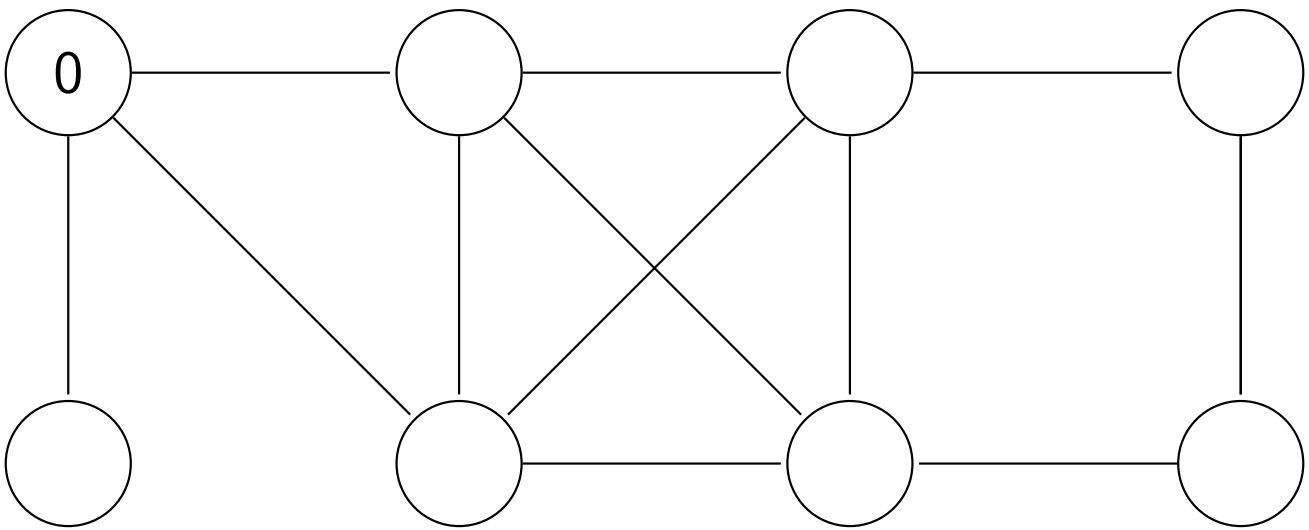


Question 5 : correction

Dans les questions précédentes, nous avons montré :

Bi-parti $\Leftrightarrow_{question4}$ Aucun cycle impair $\Leftrightarrow_{question2}$ aucune paire de sommets voisins de même profondeur

On choisit un nœud au hasard (ici en haut à gauche) et on réalise un parcours en largeur, en annotant les profondeurs,

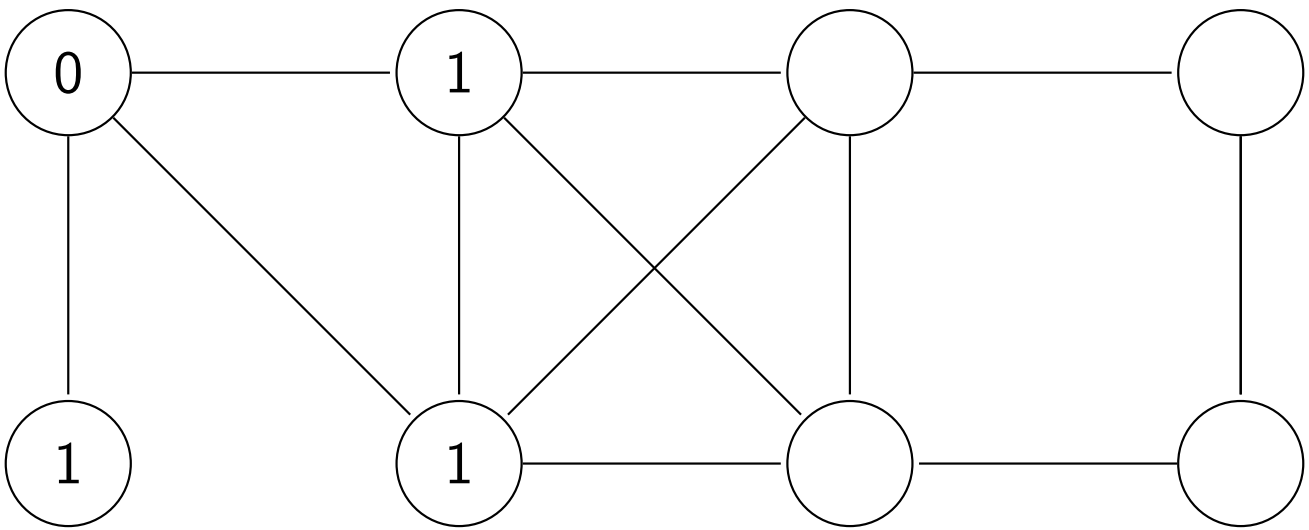


Question 5 : correction

Dans les questions précédentes, nous avons montré :

Bi-parti $\Leftrightarrow_{question4}$ Aucun cycle impair $\Leftrightarrow_{question2}$ aucune paire de sommets voisins de même profondeur

On choisit un nœud au hasard (ici en haut à gauche) et on réalise un parcours en largeur, en annotant les profondeurs,

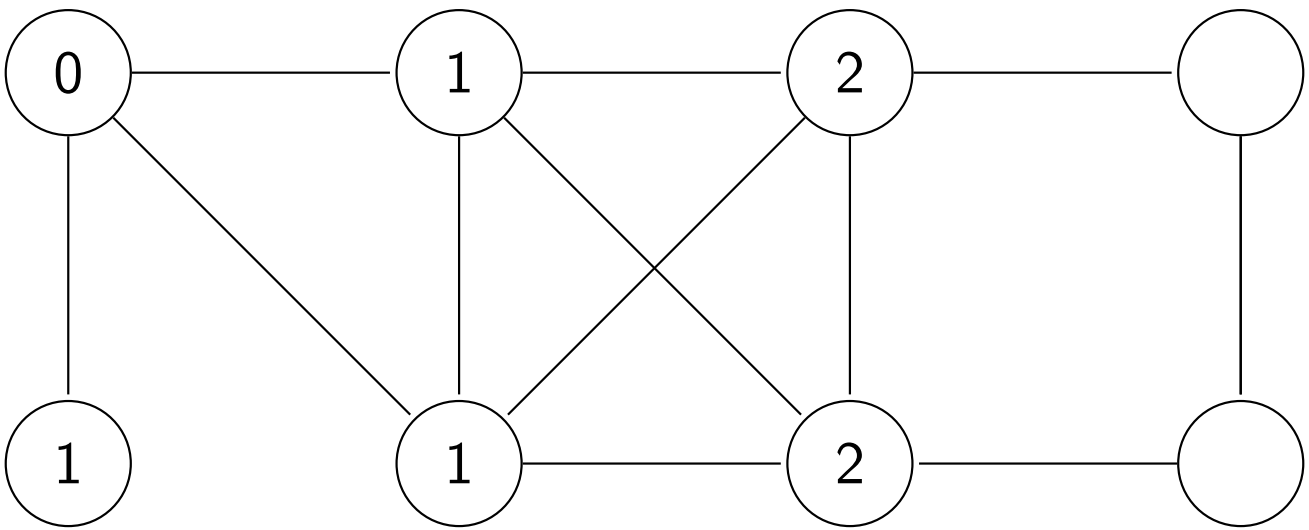


Question 5 : correction

Dans les questions précédentes, nous avons montré :

Bi-parti $\Leftrightarrow_{question4}$ Aucun cycle impair $\Leftrightarrow_{question2}$ aucune paire de sommets voisins de même profondeur

On choisit un nœud au hasard (ici en haut à gauche) et on réalise un parcours en largeur, en annotant les profondeurs,

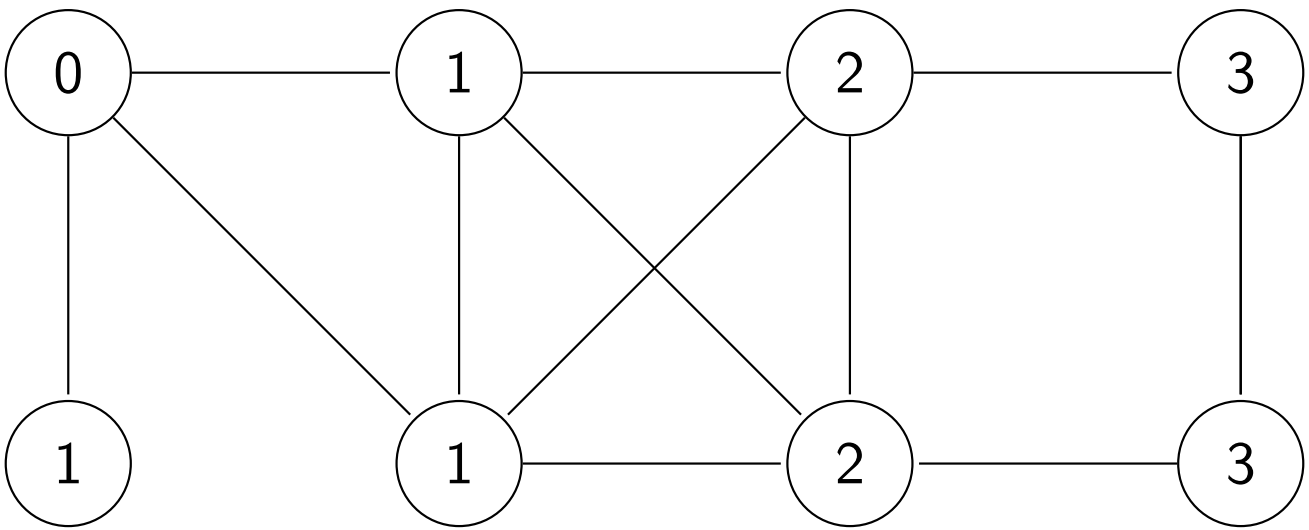


Question 5 : correction

Dans les questions précédentes, nous avons montré :

Bi-parti $\Leftrightarrow_{question4}$ Aucun cycle impair $\Leftrightarrow_{question2}$ aucune paire de sommets voisins de même profondeur

On choisit un nœud au hasard (ici en haut à gauche) et on réalise un parcours en largeur, en annotant les profondeurs,

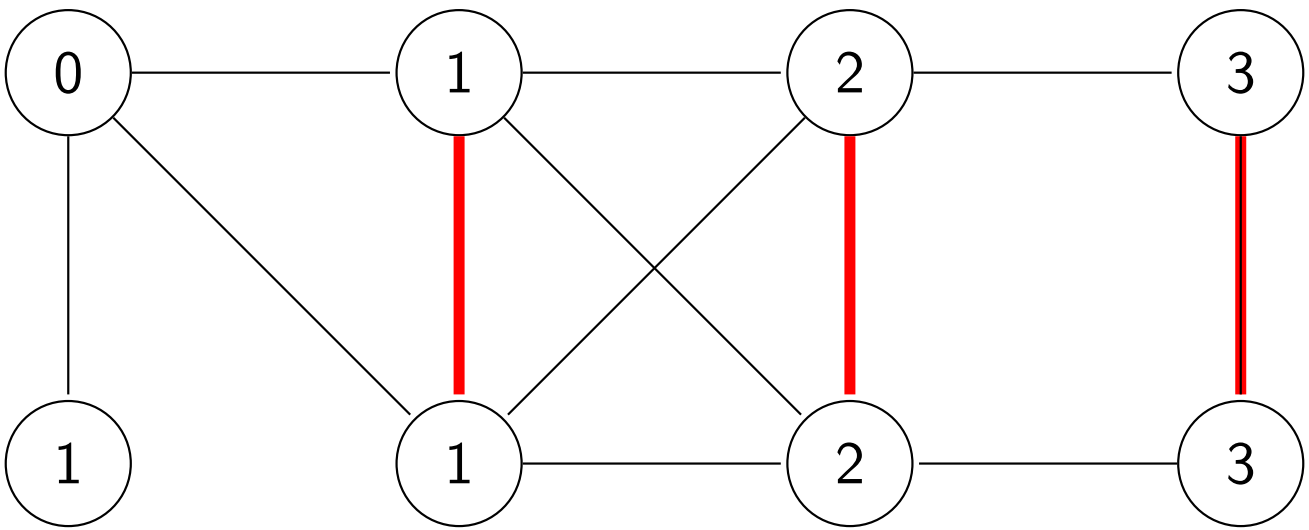


Question 5 : correction

Dans les questions précédentes, nous avons montré :

Bi-parti $\Leftrightarrow_{question4}$ Aucun cycle impair $\Leftrightarrow_{question2}$ aucune paire de sommets voisins de même profondeur

On choisit un nœud au hasard (ici en haut à gauche) et on réalise un parcours en largeur, en annotant les profondeurs,

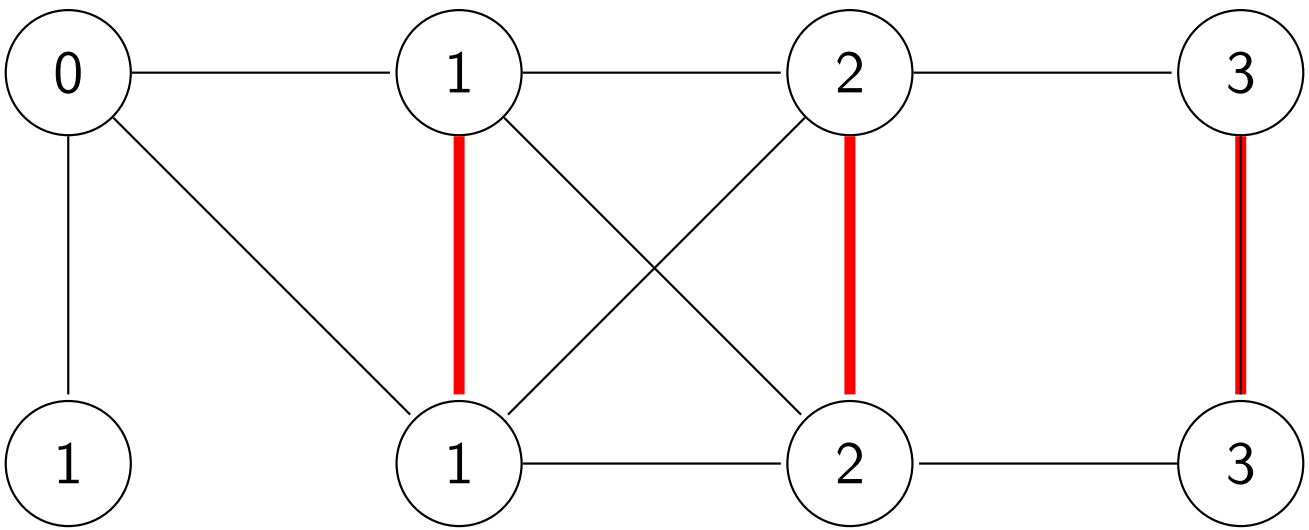


Question 5 : correction

Dans les questions précédentes, nous avons montré :

Bi-parti $\Leftrightarrow_{question4}$ Aucun cycle impair $\Leftrightarrow_{question2}$ aucune paire de sommets voisins de même profondeur

On choisit un nœud au hasard (ici en haut à gauche) et on réalise un parcours en largeur, en annotant les profondeurs,



Les trois arêtes en gras sont la preuve que le graphe n'est pas biparti.

Plan

Question 1

Le coloriage de graphe consiste à attribuer une couleur à chacun de ses sommets de manière à ce que deux sommets reliés par une arête soient de couleurs différentes.

Un graphe 2-coloriable est un graphe pouvant être colorié avec seulement 2 couleurs.

Question : Quel est le lien avec l'exercice précédent ? Justifiez votre réponse.

Question 1 : correction

Théorème : un graphe est 2-coloriable si et seulement si il est biparti.

Question 1 : correction

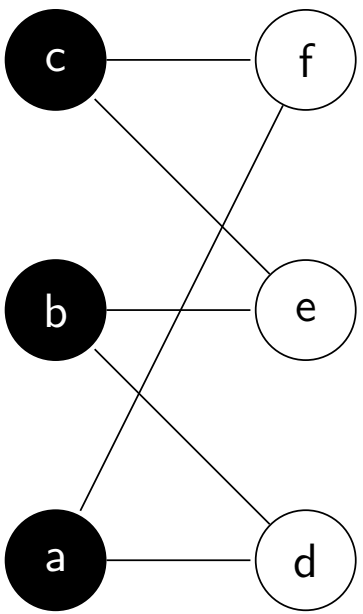
Théorème : un graphe est 2-coloriable si et seulement si il est biparti.

$\Longrightarrow$ on partitionne V en $V_1 \cup V_2$ tel que V_1 regroupe tous les noeuds blancs et V_2 tous les noirs ($V_1 \cap V_2 = \emptyset$). Toute arête connecte un noeud blanc (donc dans $V1$) à un noeud noir (dans V_2).

Question 1 : correction

Théorème : un graphe est 2-coloriable si et seulement si il est biparti.

$\Longrightarrow$ on partitionne V en $V_1 \cup V_2$ tel que V_1 regroupe tous les noeuds blancs et V_2 tous les noirs ($V_1 \cap V_2 = \emptyset$). Toute arête connecte un noeud blanc (donc dans $V1$) à un noeud noir (dans V_2).

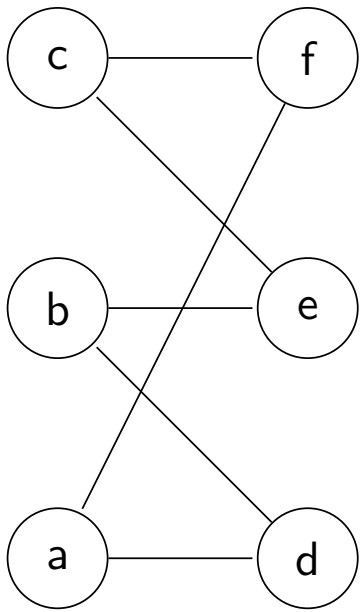


Question 1 : correction

$\Longleftarrow$ il suffit de choisir le blanc pour V_1 et le noir pour V_2 et comme toutes les arêtes sont dans $V_1 \times V_2$, on est bon.

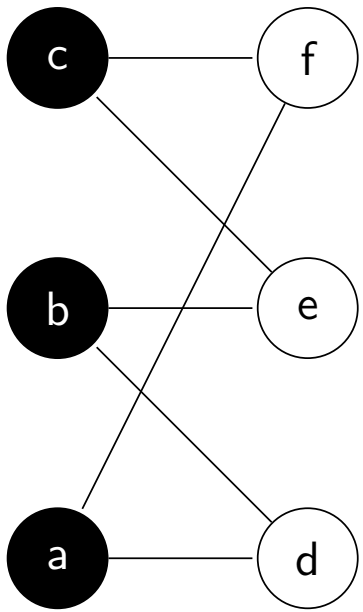
Question 1 : correction

$\Longleftarrow$ il suffit de choisir le blanc pour V_1 et le noir pour V_2 et comme toutes les arêtes sont dans $V_1 \times V_2$, on est bon.



Question 1 : correction

$\Longleftarrow$ il suffit de choisir le blanc pour V_1 et le noir pour V_2 et comme toutes les arêtes sont dans $V_1 \times V_2$, on est bon.



Question 2

Nous souhaitons écrire un algorithme, inspiré du parcours en **profondeur** vu en cours, qui prend en entrée un graphe `(V,E)` et qui renvoie un couple `(result,color)` où `result` est `True` si le graphe est coloriable, `False` sinon et `color` est un dictionnaire associant une couleur 0 ou 1 à chaque sommet.

Cet algorithme devra **s'arrêter au plus tôt** lorsque le graphe n'est pas 2-coloriable.

Proposez une version **itérative** et une version **récursive**.

Question 2 : correction

On réalise un parcours en profondeur depuis une racine r quelconque. La racine est coloré en blanc (0) ou en noir (1). Puis on alterne de couleur de voisin en voisin et on s'arrête dès qu'un mauvais coloriage est détecté.

Question 2 : Exploration en profondeur (version récursive)

```
def DFS_rec(V,E,n,t):     # n : le noeud courant
  visited[n] = True
  if n==t: return True
  for v in neighbours(n,E):
    if not v in visited:
      if DFS_rec(V,E,v,t):
        return True
  return False
```

Question 2 : Exploration en profondeur (version itérative)

```
def DFS_iter(V,E,s,t):
    lnext = [s]                  # la pile
    reached = { s: True }        # eviter des ajouts multiples
    while len(lnext)>0:
      n = pop_end(lnext)
      if n==t:
        return True
      for v in neighbours(n,E):
        if not v in reached:
          reached[v] = True
          add_end(v,lnext)       # recursion -> ajout dans pile
    return False
```

Question 2 : correction

Pour commencer, on réalise un parcours en profondeur depuis une racine r quelconque. La racine est coloré en blanc (0) ou en noir (1). Puis on alterne de couleur de voisin en voisin et on s'arrête dès qu'un mauvais coloriage est détecté.

Algorithme itératif :

```
def coloring2_iter(V,E,s=V[0]):
    color = {s: 0}
    lnext = [s]
    while len(lnext)>0:
        n = pop_end(lnext)
        for v in neighbours(n,E):
            if not v in color:
                add_end(v,lnext)
                color[v] = 1-color[n]
            elif color[v] == color[n]:
                return False, color
    return True, color
```

Question 2 : correction

Algorithme récursif :

```
# recursive function called by coloring2
def coloring2_rec(V,E,n,color):
  for v in neighbours(n,E):
    if not v in color:
      color[v] = 1-color[n]
      if not coloring2_rec(V,E,v,color):
        return False
    elif color[v] == color[n]:
      return False
  return True

# the calling function for recursive version
def coloring2(V,E,s=V[0]):
   color = {s:0}
   colorable = coloring2_rec(V,E,s,color)
   return colorable, color
```

Question 3

Question : Est ce que votre algorithme est correct, c'est-à-dire :

1. Quand il retourne un coloriage, ce 2-coloriage est-il correct ?
2. Quand il retourne `False`, est-on certain que le graphe n'est pas 2-coloriable ?

Question 3 : correction

1. Si l'algo trouve un coloriage alors **il a déjà testé toutes les arêtes**, car tous les noeuds sont visités et pour chaque voisin soit il n'est pas encore coloré et on lui donne une couleur différente (qui ne bougera plus), soit il est déjà coloré et on vérifie le coloriage.

Question 3 : correction

1. Si l'algo trouve un coloriage alors **il a déjà testé toutes les arêtes**, car tous les noeuds sont visités et pour chaque voisin soit il n'est pas encore coloré et on lui donne une couleur différente (qui ne bougera plus), soit il est déjà coloré et on vérifie le coloriage.
2. Si l'algo ne trouve pas de coloriage, l'algo a trouvé 2 noeuds voisins x et y colorés pareil !

Question 3 : correction

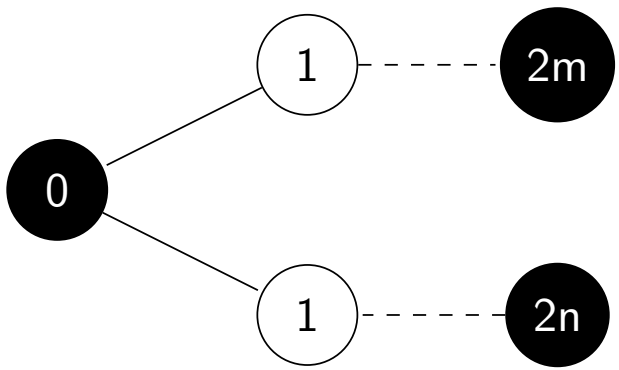
1. Si l'algo trouve un coloriage alors **il a déjà testé toutes les arêtes**, car tous les noeuds sont visités et pour chaque voisin soit il n'est pas encore coloré et on lui donne une couleur différente (qui ne bougera plus), soit il est déjà coloré et on vérifie le coloriage.
2. Si l'algo ne trouve pas de coloriage, l'algo a trouvé 2 noeuds voisins x et y colorés pareil !

On pourrait imaginer qu'un autre parcours aurait conduit à ne pas colorer ces voisins de la même couleur, ce qui voudrait dire que **le graphe est en fait bi-coloriable ?**.

Question 3 : correction

Considérons l'arbre de visite/coloriage réalisé avant l'arrêt de l'algo sur l'arête (x, y).
Dans cet arbre, la couleur d'un sommet indique la parité du chemin depuis la racine (la parité de la profondeur dans cet arbre mais pas nécessairement celle du graphe).

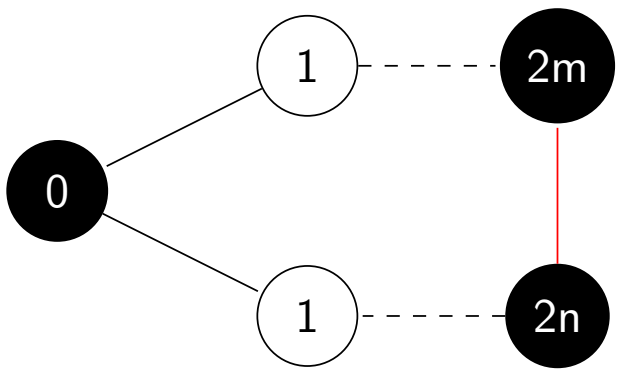
Si on a deux sommets de même couleur, cela veut juste dire qu'ils ont la même parité dans cet arbre.



Question 3 : correction

Considérons l'arbre de visite/coloriage réalisé avant l'arrêt de l'algo sur l'arête (x, y).
Dans cet arbre, la couleur d'un sommet indique la parité du chemin depuis la racine (la parité de la profondeur dans cet arbre mais pas nécessairement celle du graphe).

Si on a deux sommets de même couleur, cela veut juste dire qu'ils ont la même parité dans cet arbre.

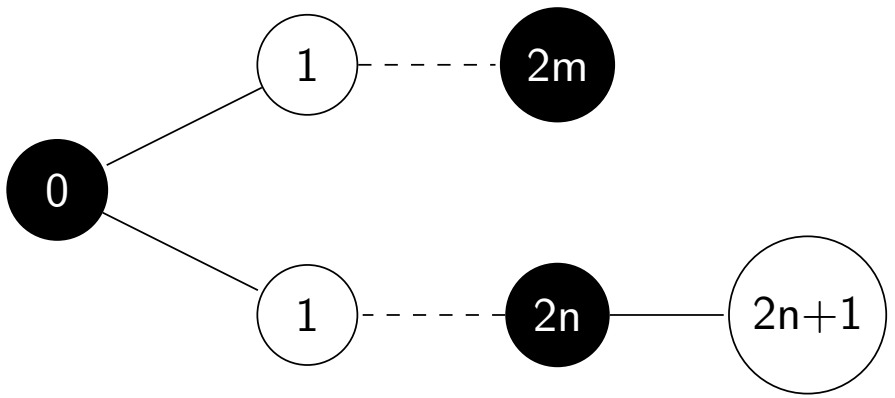


Cycle impair $\Longrightarrow$ graphe non-biparti $\Longrightarrow$ graphe non bi-coloriable

Question 3 : correction

Considérons l'arbre de visite/coloriage réalisé avant l'arrêt de l'algo sur l'arête (x, y).
Dans cet arbre, la couleur d'un sommet indique la parité du chemin depuis la racine (la parité de la profondeur dans cet arbre mais pas nécessairement celle du graphe).

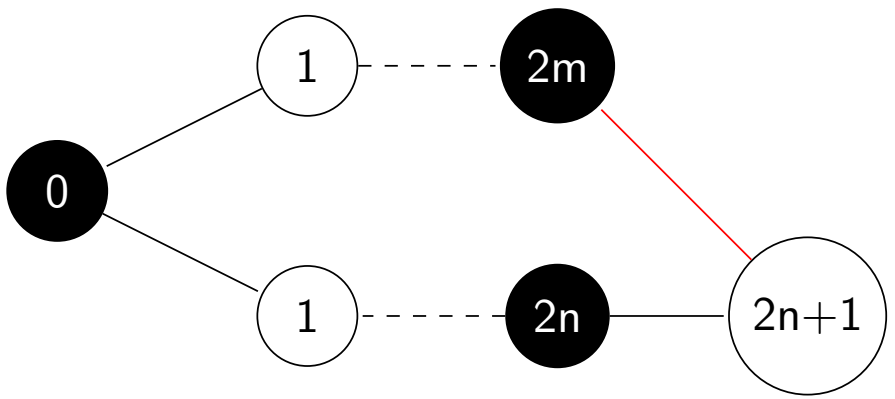
Si on a deux sommets de même couleur, cela veut juste dire qu'ils ont la même parité dans cet arbre.



Question 3 : correction

Considérons l'arbre de visite/coloriage réalisé avant l'arrêt de l'algo sur l'arête (x, y).
Dans cet arbre, la couleur d'un sommet indique la parité du chemin depuis la racine (la parité de la profondeur dans cet arbre mais pas nécessairement celle du graphe).

Si on a deux sommets de même couleur, cela veut juste dire qu'ils ont la même parité dans cet arbre.

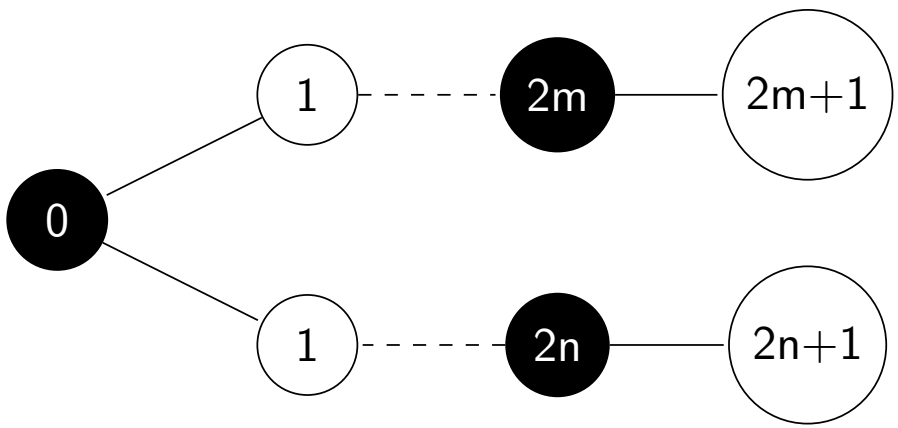


Cycle paire

Question 3 : correction

Considérons l'arbre de visite/coloriage réalisé avant l'arrêt de l'algo sur l'arête (x, y).
Dans cet arbre, la couleur d'un sommet indique la parité du chemin depuis la racine (la parité de la profondeur dans cet arbre mais pas nécessairement celle du graphe).

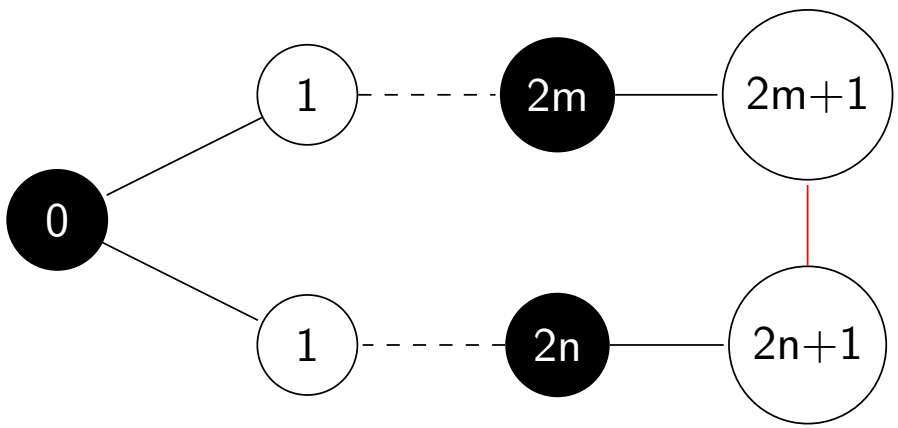
Si on a deux sommets de même couleur, cela veut juste dire qu'ils ont la même parité dans cet arbre.



Question 3 : correction

Considérons l'arbre de visite/coloriage réalisé avant l'arrêt de l'algo sur l'arête (x, y).
Dans cet arbre, la couleur d'un sommet indique la parité du chemin depuis la racine (la parité de la profondeur dans cet arbre mais pas nécessairement celle du graphe).

Si on a deux sommets de même couleur, cela veut juste dire qu'ils ont la même parité dans cet arbre.



Cycle impair $\Longrightarrow$ graphe non-biparti $\Longrightarrow$ graphe non bi-coloriable

Question 3 : correction

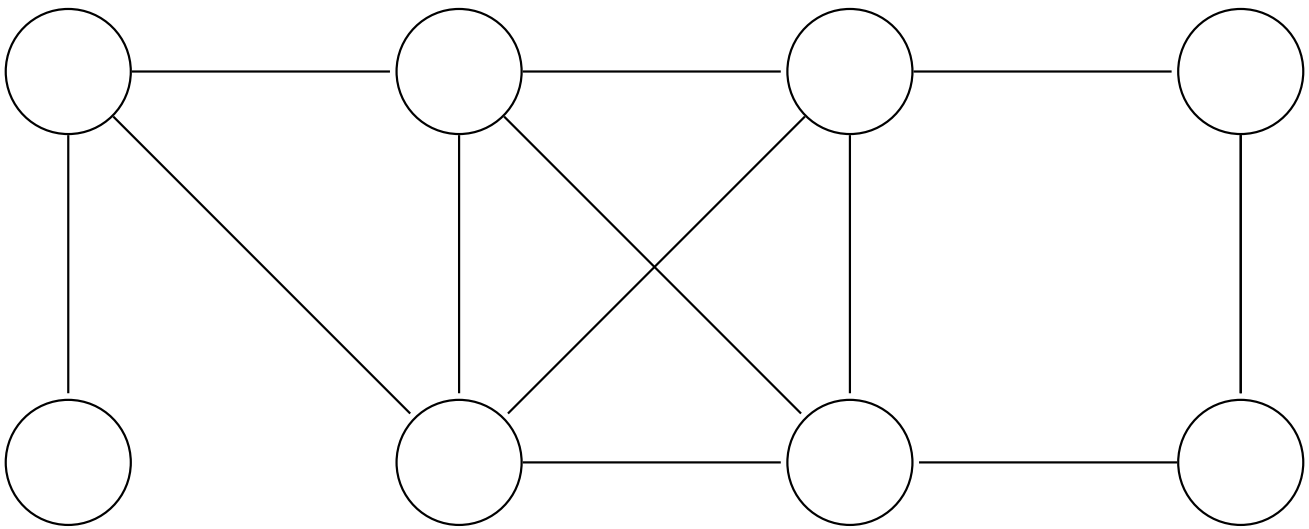
De manière similaire au raisonnement en question 2 de l'exercice 1 :

x et y ont un premier ancêtre z en commun (éventuellement la racine) à partir duquel on peut former un cycle impair de taille $(depth(x) + depth(y) - 2\ depth(z)) + 1$ en ajoutant l'arête (x, y) au sous-arbre de z.

Le graphe n'est donc pas biparti et pas 2-coloriable.

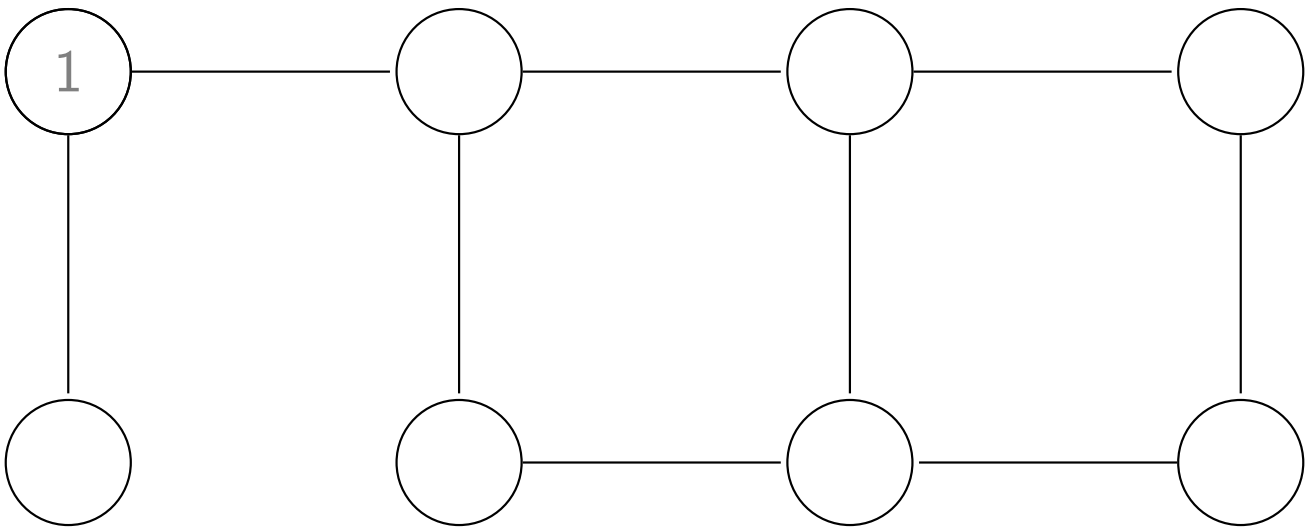
Question 4

Testez votre algorithme sur le graphe ci-dessus. Est-il 2-coloriable ?



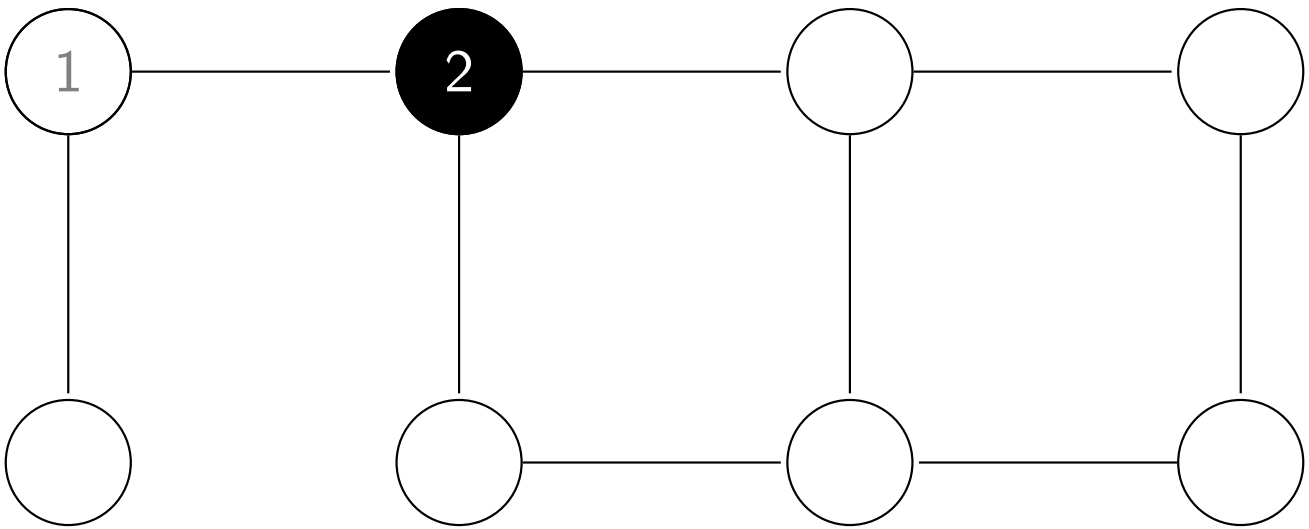
Question 4 : correction

On choisit un noeud au hasard (ici en haut à gauche) et on réalise un parcours en profondeur, en annotant les couleurs, on obtient le graphe suivant, les numéros correspondent à l'ordre du parcours :



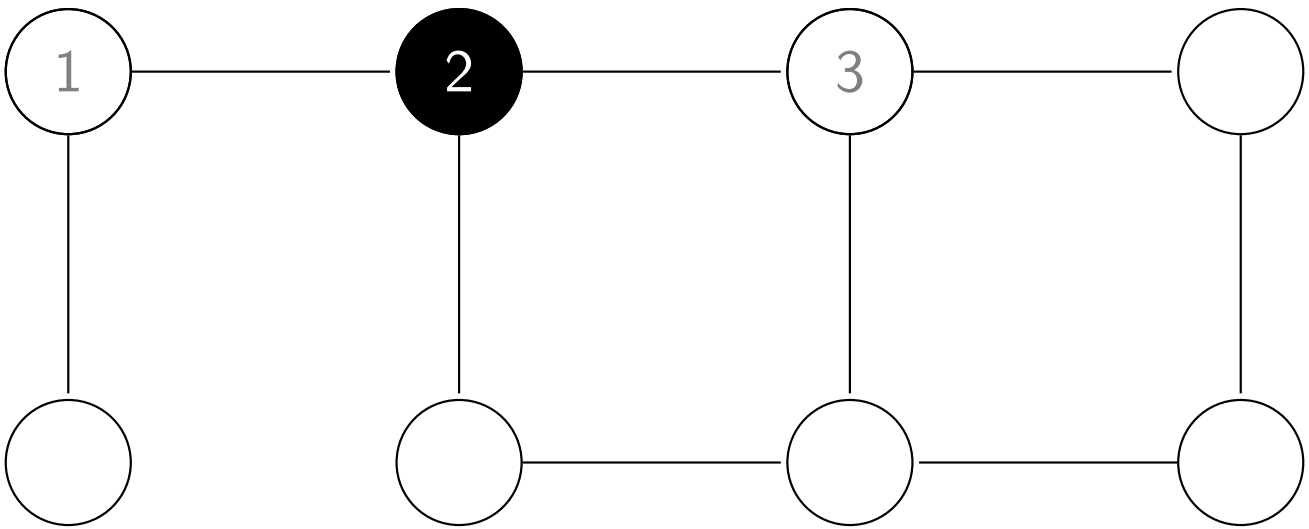
Question 4 : correction

On choisit un noeud au hasard (ici en haut à gauche) et on réalise un parcours en profondeur, en annotant les couleurs, on obtient le graphe suivant, les numéros correspondent à l'ordre du parcours :



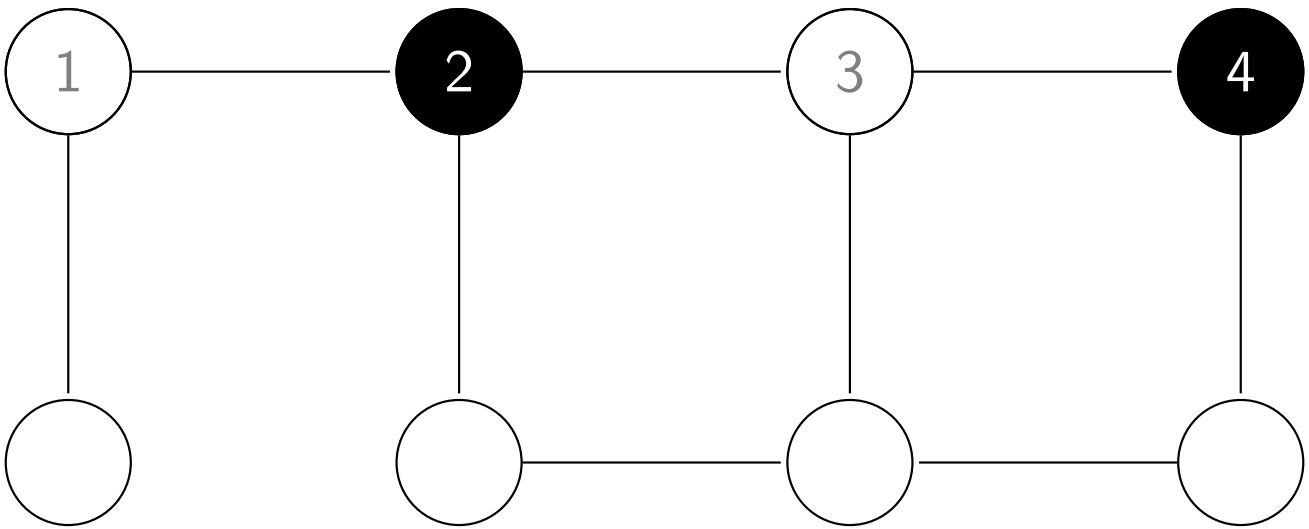
Question 4 : correction

On choisit un noeud au hasard (ici en haut à gauche) et on réalise un parcours en profondeur, en annotant les couleurs, on obtient le graphe suivant, les numéros correspondent à l'ordre du parcours :



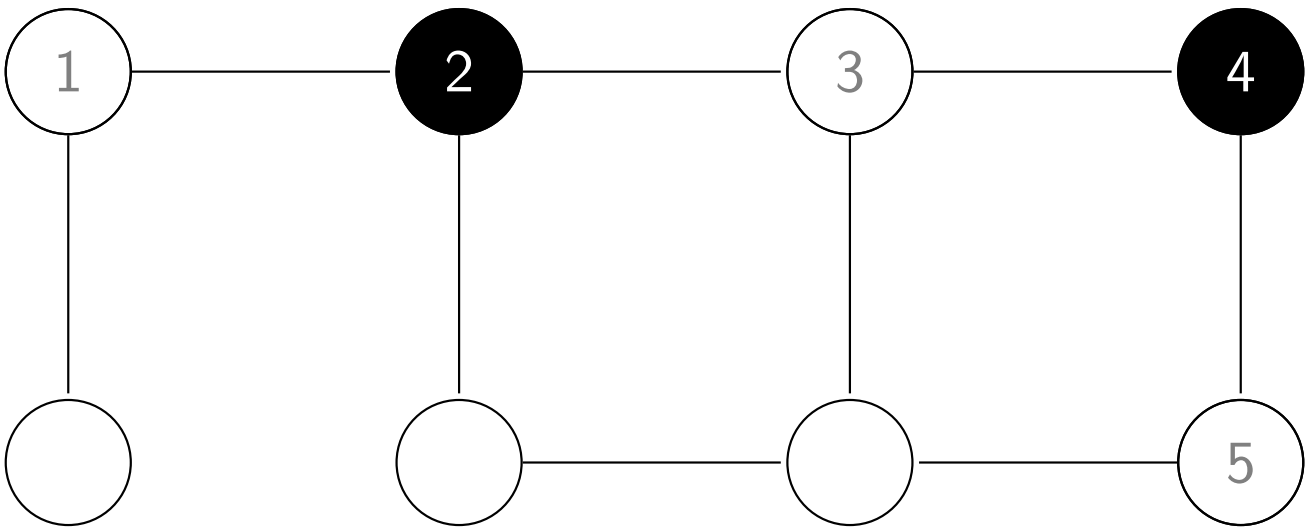
Question 4 : correction

On choisit un noeud au hasard (ici en haut à gauche) et on réalise un parcours en profondeur, en annotant les couleurs, on obtient le graphe suivant, les numéros correspondent à l'ordre du parcours :



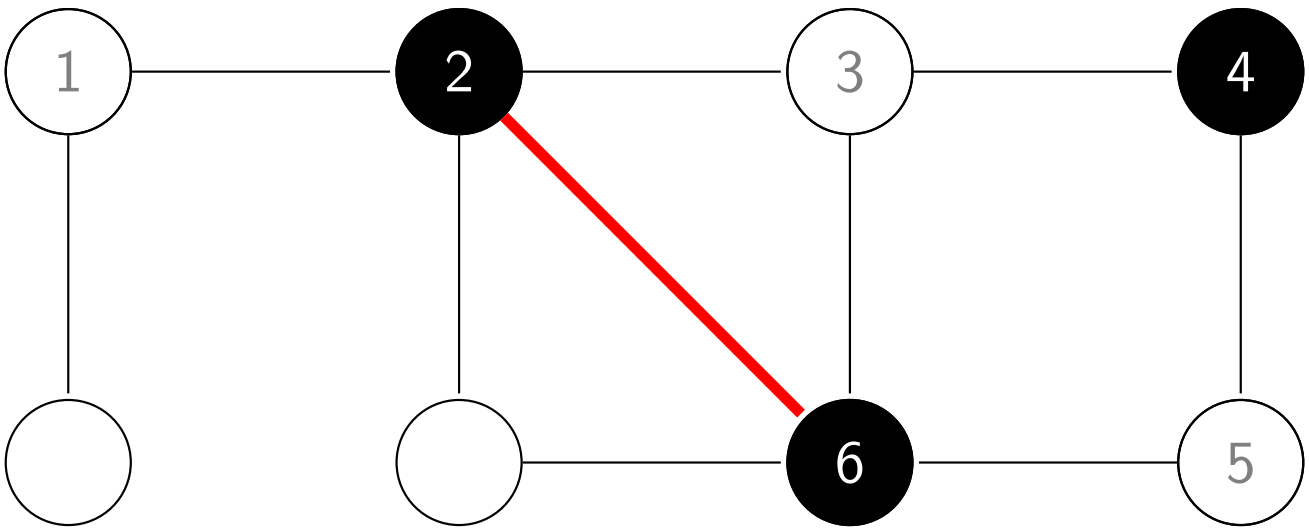
Question 4 : correction

On choisit un noeud au hasard (ici en haut à gauche) et on réalise un parcours en profondeur, en annotant les couleurs, on obtient le graphe suivant, les numéros correspondent à l'ordre du parcours :



Question 4 : correction

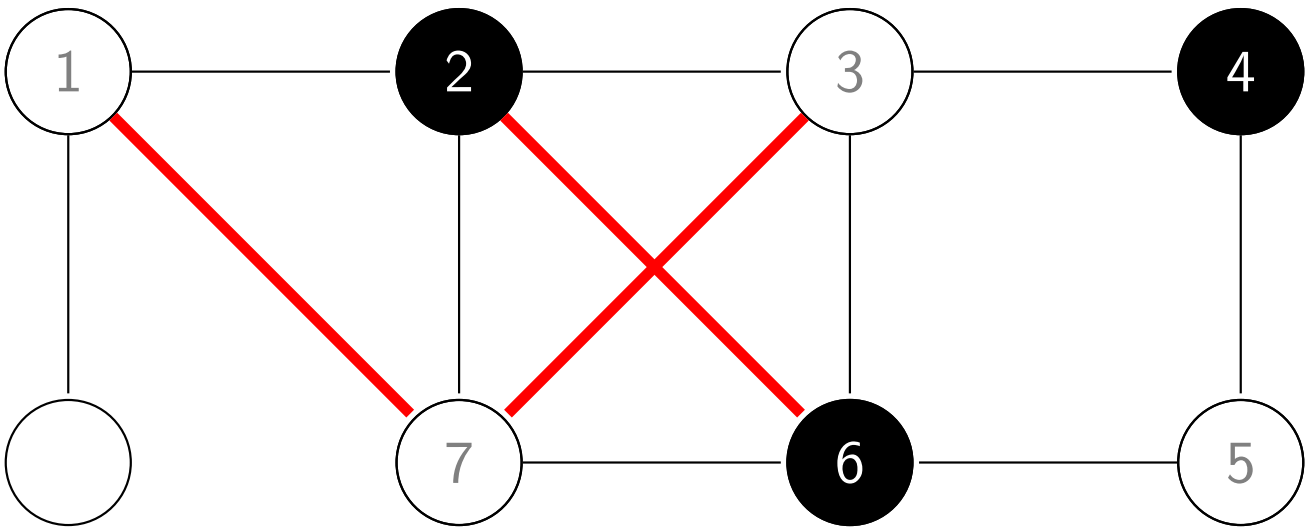
On choisit un noeud au hasard (ici en haut à gauche) et on réalise un parcours en profondeur, en annotant les couleurs, on obtient le graphe suivant, les numéros correspondent à l'ordre du parcours :



Ce graphe n'est pas 2-coloriable à cause des arêtes en gras qui relient des sommets de même couleur.

Question 4 : correction

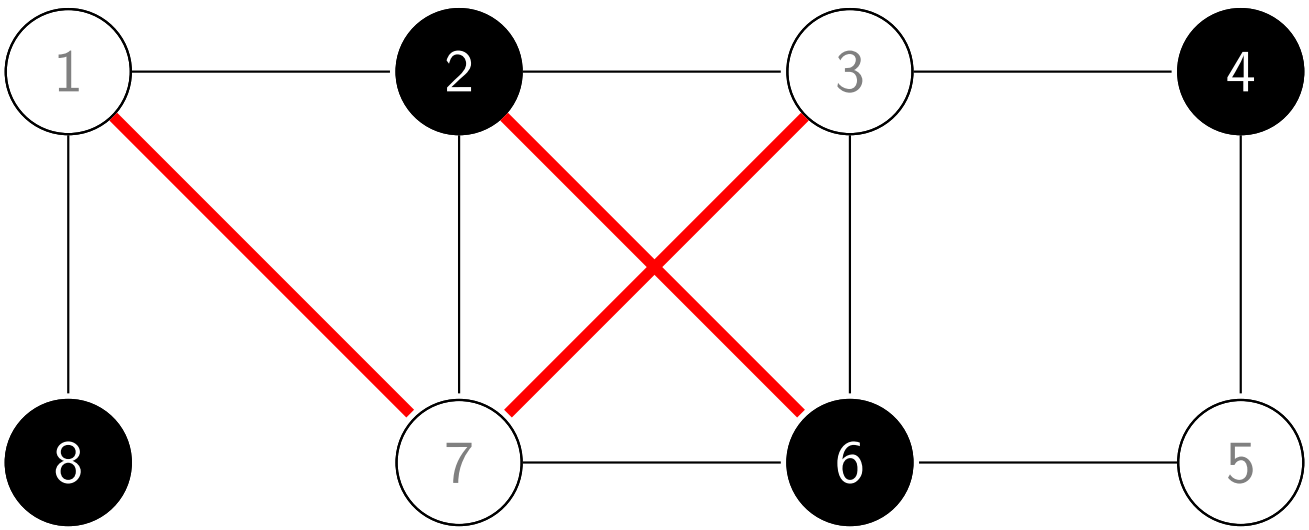
On choisit un noeud au hasard (ici en haut à gauche) et on réalise un parcours en profondeur, en annotant les couleurs, on obtient le graphe suivant, les numéros correspondent à l'ordre du parcours :



Ce graphe n'est pas 2-coloriable à cause des arêtes en gras qui relient des sommets de même couleur.

Question 4 : correction

On choisit un noeud au hasard (ici en haut à gauche) et on réalise un parcours en profondeur, en annotant les couleurs, on obtient le graphe suivant, les numéros correspondent à l'ordre du parcours :



Ce graphe n'est pas 2-coloriable à cause des arêtes en gras qui relient des sommets de même couleur.

Plan

Question 1

Prouvez le théoreme suivant :
Dans un graphe orienté sans circuit (c'est à dire un DAG – *Directed Acyclic Graph*) il existe au moins un sommet sans arcs entrants.

Question 1 : correction

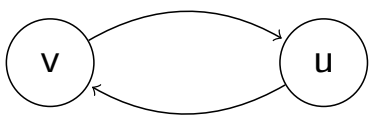
Soit G un DAG. Supposons que tous ses sommets aient au moins un arc entrant.

- Soit v un sommet de G. Il a un arc entrant (u, v). On recule.

Question 1 : correction

Soit G un DAG. Supposons que tous ses sommets aient au moins un arc entrant.

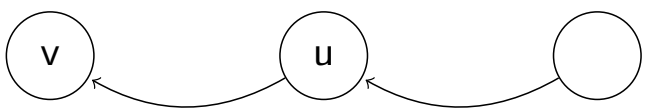
- Soit v un sommet de G. Il a un arc entrant (u, v). On recule.
- On considère maintenant le sommet u. Il a un arc entrant (x, u). Si $x = v$, on a un circuit.



Question 1 : correction

Soit G un DAG. Supposons que tous ses sommets aient au moins un arc entrant.

- Soit v un sommet de G. Il a un arc entrant (u, v). On recule.
- On considère maintenant le sommet u. Il a un arc entrant (x, u). Si $x = v$, on a un circuit. Sinon, on recule à nouveau.



Question 1 : correction

Soit G un DAG. Supposons que tous ses sommets aient au moins un arc entrant.

- Soit v un sommet de G. Il a un arc entrant (u, v). On recule.
- On considère maintenant le sommet u. Il a un arc entrant (x, u). Si $x = v$, on a un circuit. Sinon, on recule à nouveau.
- Nous remontons ainsi les sommets jusqu'à croiser un sommet déjà rencontré.

Question 1 : correction

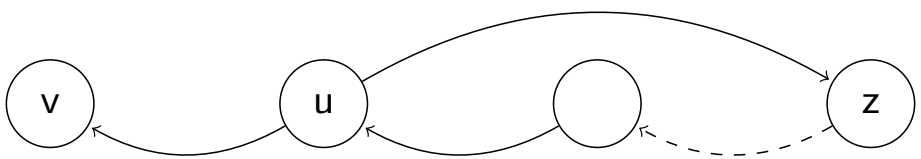
Soit G un DAG. Supposons que tous ses sommets aient au moins un arc entrant.

- Soit v un sommet de G. Il a un arc entrant (u, v). On recule.
- On considère maintenant le sommet u. Il a un arc entrant (x, u). Si $x = v$, on a un circuit. Sinon, on recule à nouveau.
- Nous remontons ainsi les sommets jusqu'à croiser un sommet déjà rencontré.
- Cette situation arrive forcément puisque tout sommet a un arc entrant : même si on élimine tous les sommets un par un, arrivé au dernier sommet z, il faudra bien qu'il ait un arc entrant (y, z) avec y un autre sommet... déjà visité !

Question 1 : correction

Soit G un DAG. Supposons que tous ses sommets aient au moins un arc entrant.

- Soit v un sommet de G. Il a un arc entrant (u, v). On recule.
- On considère maintenant le sommet u. Il a un arc entrant (x, u). Si $x = v$, on a un circuit. Sinon, on recule à nouveau.
- Nous remontons ainsi les sommets jusqu'à croiser un sommet déjà rencontré.
- Cette situation arrive forcément puisque tout sommet a un arc entrant : même si on élimine tous les sommets un par un, arrivé au dernier sommet z, il faudra bien qu'il ait un arc entrant (y, z) avec y un autre sommet... déjà visité !



Question 1 : correction

Soit G un DAG. Supposons que tous ses sommets aient au moins un arc entrant.

- Soit v un sommet de G. Il a un arc entrant (u, v). On recule.
- On considère maintenant le sommet u. Il a un arc entrant (x, u). Si $x = v$, on a un circuit. Sinon, on recule à nouveau.
- Nous remontons ainsi les sommets jusqu'à croiser un sommet déjà rencontré.
- Cette situation arrive forcément puisque tout sommet a un arc entrant : même si on élimine tous les sommets un par un, arrivé au dernier sommet z, il faudra bien qu'il ait un arc entrant (y, z) avec y un autre sommet... déjà visité !
- Cela signifie la présence d'un circuit, d'où la contradiction.