TD11 - Algorithmique

Code	LU2IN003
Type de cours	Travaux dirigés
Complété ?	☐
Jour du cours	@19/04/2023

`PINHO FERNANDES Enzo - L2 Mono-Info S4 - GR6`

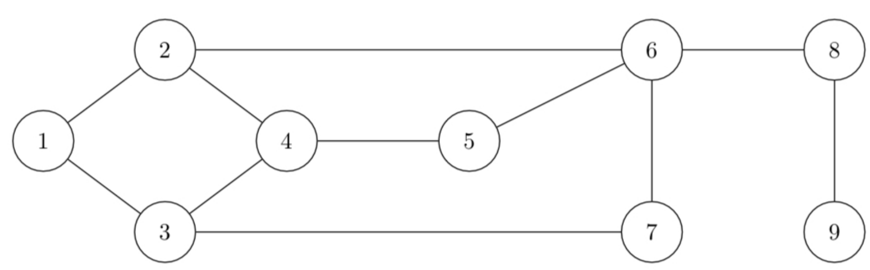
TD11 : Parcours en profondeur

Exercice 1 : Graphe non orienté : exercice de base

📖 **Enoncé :**

Exercice 1 – Graphe non orienté : exercice de base

On considère le graphe non orienté $G_1 = (V_1, E_1)$:



Question 1
Pour chacun des parcours génériques de G_1 suivants, dire s'il est ou non un parcours en profondeur : $(5, 6, 8, 9, 7, 3, 1, 2, 4)$, $(8, 6, 9, 7, 5, 2, 4, 3, 1)$, $(4, 2, 1, 5, 3, 7, 6, 8, 9)$, $(4, 2, 6, 8, 9, 5, 7, 3, 1)$
Justifier les réponses négatives.

Question 2
Donner trois parcours en profondeur de G_1, l'un partant du sommet 1, un autre du sommet 9 et un troisième du sommet 5.

Question 3
On considère le parcours $L = (3, 7, 6, 8, 9, 5, 4, 2, 1)$ de G_1. Dire quel est le dernier sommet ouvert de chaque sous-parcours de L. Le parcours L est-il un parcours en profondeur ?

💡 **Réponses :**

▼ **Question 1-??? :**

- Parcours en profondeur : Soit $(v_1, ..., v_k), v_{k+1}$ doit être un sommet adjacent au **DERNIER** sommet ouvert.
    - En comparaison, la parcours en largeur a la même définition, à la différence que le sommet est adjacent au **PREMIER** sommet ouvert.
- $(4, 5, 6, 8, 9, 2, 1, 3, 7)$ marche.
- $(3, 7, 6, 8, 2, 1, 4, 5, 9)$ ne marche pas, au lieu de 2 il devrait y avoir 9.

Parcours	Dernier sommet ouvert
(2)	2
$(2, 1)$	1
$(2, 1, 3)$	3
$(2, 1, 3, 7)$	7
$(2, 1, 3, 7, 6)$	6
$(2, 1, 3, 7, 6, 8)$	8
$(2, 1, 3, 7, 6, 8, 9)$	6

▼ Exercice 3 : Algorithme de calcul d’un parcours en profondeur d’un graphe non orienté

📖 **Enoncé :**

Exercice 3 – Algorithme de calcul d'un parcours en profondeur d'un graphe non orienté

On rappelle l'algorithme de calcul d'un parcours en profondeur vu en cours :

Require: Un graphe non orienté connexe $G = (V, E)$, un sommet s

Ensure: Un parcours en profondeur L d'origine s

```
function DFS(G, s)
  visite[s] := True, L := (s)
  for all arete {s, u} ∈ E do
    if not visite[u] then
      L := L+DFS(G, u)
    end if
  end for
  return L
end function
```

$visite$ est un tableau de booléens sur les sommets dont les cases sont initialisées à $Faux$.

Question 1

Exécuter DFS(G_1, 6). Pour cela, donner l'arbre des exécutions et le parcours en profondeur obtenu, ainsi que l'évolution de l'ensemble des sommets visités. Le cas échéant, prendre en priorité le sommet de plus petit numéro.

Question 2

Donnez les valeurs pre et $post$ associées au parcours en profondeur obtenu à la question précédente.

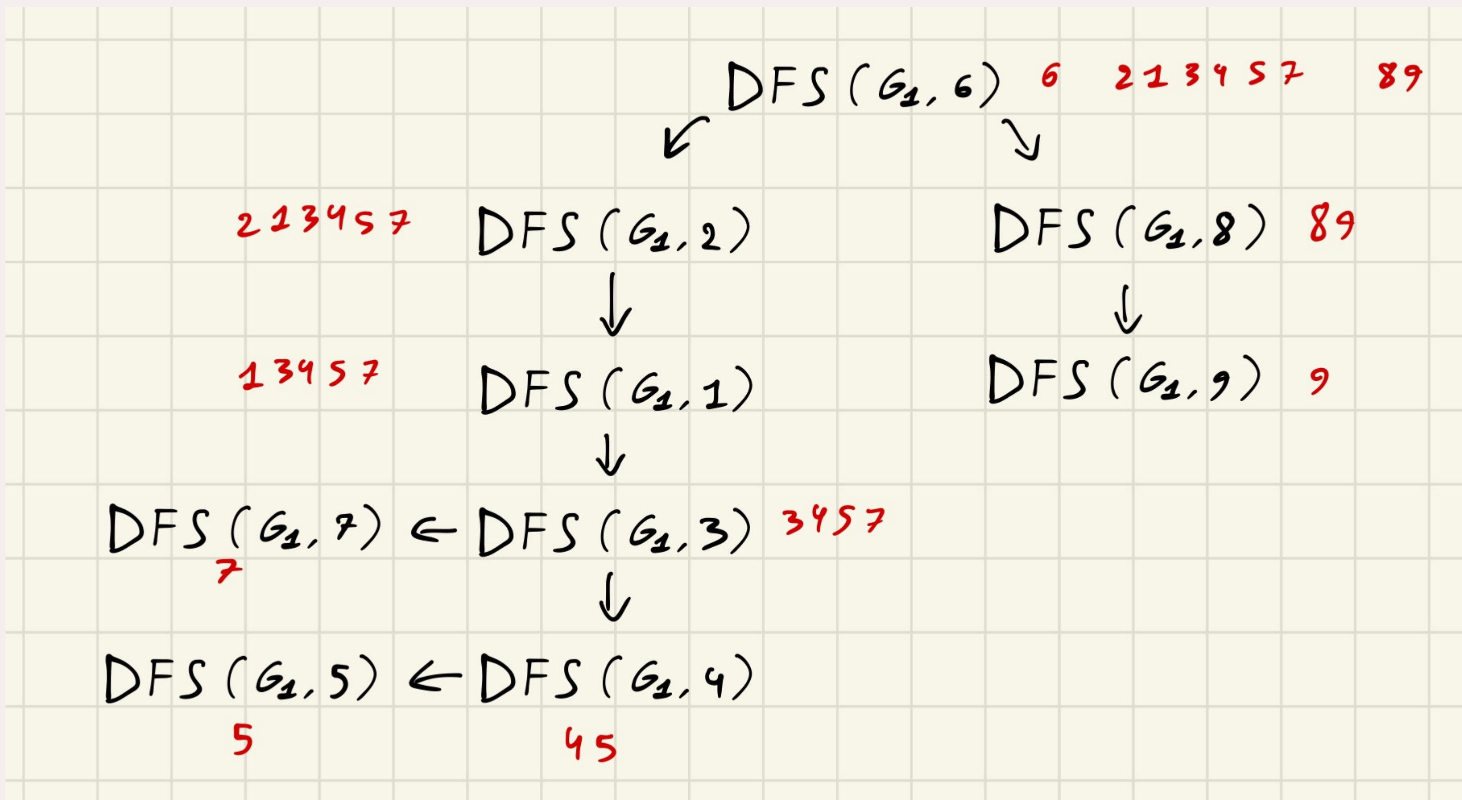
Question 3

1. Que peut-on dire dans le cas général des intervalles $[pre(u), post(u)]$ et $[pre(v), post(v)]$ si (u, v) est un arc de liaison ?
2. Si maintenant $\{u, v\}$ est une arête du graphe qui ne correspond pas à un arc de liaison, mais telle qu'il y a un chemin de u à v dans le graphe de laison en profondeur $\mathcal{A}^(L)$. Que peut-on dire dans le cas général des intervalles $[pre(u), post(u)]$ et $[pre(v), post(v)]$?

💡 **Réponses :**

▼ **Question 1 :**

- Chaque sommet est visité une fois (avec `visite[u] = True`)
- Appel récursif sur les sommets adjacents (parcours en profondeur ⇔ arbre des appels)



▼ **Question 2 :**

u	$pre(u)$ (ouverture)	$post(u)$ (fermeture)
6	1	18 ($9 * 2$)
$$2$$	2	13
1	3	12
3	4	11
4	5	$$8$$
5	6	7
7	9	10
8	14	17
9	15	16

▼ **Question 3 :**

- Définition : arc de liaison, arc $\in$ parcours (e.g $(6, 2)$, $(1, 3)$, $(8, 9)$)
- (u, v) arc de liaison $u \to v$
  - $[pre(v), post(v)] \subset [pre(u), post(u)]$
  - Autrement dit, u est ouvert avant v et u est fermé avant v.

▼ Exercice 4 : ???

📖 **Enoncé :**

Exercice 4

On rappelle l'algorithme de construction d'un parcours en largeur :

Require: Un graphe non orienté connexe $G = (V, E)$, un sommet s
Ensure: Un parcours en largeur L d'origine s, les valeurs $dist_s(u), u \in V$

```
for all u ∈ V do
    dist_s(u) := +∞
end for
L := (), Enfiler (F,s), dist_s(s) := 0
while not FileVide(F) do
    u :=Défiler(F), L := L + (u)
    for all {u, v} ∈ E do
        if dist_s(v) = +∞ then
            Enfiler(F,v), dist_s(v) = dist_s(u) + 1
        end if
    end for
end while
```

Question 1

Appliquer cet algorithme au graphe G_1 en partant du sommet $s = 4$. Préciser, à chaque itération, le sommet u retiré à F, le sous-parcours L, la valeur de la file F, la valeur des $dist_s(v)$ pour $v \in V_1$. Les valeurs de L, de F et de $dist_s$ sont celles obtenues à chaque itération en fin du corps de boucle.

💡 **Réponses :**

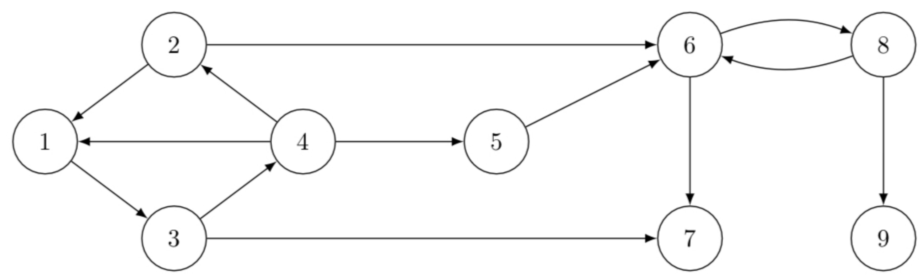
▼ **Question 1 :**

- Complexité algorithme (exo 3)
  - Initialisation (`visite[u]`) en $\Theta(n)$, une fois pour chaque sommet.
  - Toutes les instructions sont en $\Theta(1)$ sauf le calcul des sommets adjacents en $cv(u)$
  - Chaque sommet est traité exactement une fois.
    - $\Rightarrow \Theta(n + \Sigma_{u \in V} \ cv(u))$ (Même que pour le parcours en largeur)
- Donc, suivant sa représentation, la complexité est de :
  - Matrice sommet-sommet : $\Theta(n^2)$
  - Matrice sommet-arête : $\Theta(m * n^2)$
  - Liste adjacente : $\Theta(n + m)$

▼ Exercice 5 : Graphe orienté, exercice de base

📖 **Enoncé :**

Question 1
On considère le graphe orienté $G_3 = (V_3, A_3)$:



Pour chaque racine de G_3, donner un parcours en profondeur de G_3.

💡 **Réponses :**

▼ **Question 1 :**

- Parcours largeur / profondeur : condition nécessaire
  - Non orienté : G connexe
  - Orienté : $\exists$? racine $= \{1, 2, 3, 4\}$
- $(3, 7, 4, 2, 1, 6, 8, 9, 5)$ vraie, avec 3 comme racine.

▼ Exercice 6 : Algorithme de calcul d'un parcours en profondeur d'un graphe orienté

📖 **Enoncé :**

Exercice 6 – Algorithme de calcul d'un parcours en profondeur d'un graphe orienté

Voici un algorithme de calcul d'un parcours en profondeur d'un graphe orienté, dans lequel on calcule aussi les valeurs de pre et $post$:

Require: Un graphe orienté $G = (V, E)$ ayant au moins une racine, une racine s
Ensure: Un parcours en profondeur L d'origine s

```
function DFS(G, s)
  visite[s] := True ; L := (s)
  cpt = cpt + 1 ; pre[s] = cpt
  for all arc (s, u) ∈ E do
    if not visite[u] then
      L := L+DFS(G, u)
    end if
  end for
  cpt = cpt + 1 ; post[s] = cpt
  return L
end function
```

$visite$ est un tableau de booléens sur les sommets dont les cases sont initialisées à $Faux$.
cpt est une variable globale entière, initialisée à 0.
pre et $post$ sont deux tableaux de n entiers, initialisés à 0.

Question 1

Exécuter DFS$(G_1, 4)$ de façon à obtenir le parcours en profondeur $(4, 5, 6, 7, 8, 9, 2, 1, 3)$. Pour cela, donner l'arbre des exécutions et le parcours en profondeur obtenu, ainsi que l'évolution de cpt et celle des tableaux $visite$, pre et $post$ à chaque ouverture et chaque fermeture d'un sommet. On rappelle qu'un sommet u est ouvert à l'appel DFS(G, u) et est fermé au moment où on dépile DFS(G, u).

Question 2

Classifier les arcs de G_3 en fonction du parcours en profondeur $(4, 5, 6, 7, 8, 9, 2, 1, 3)$.

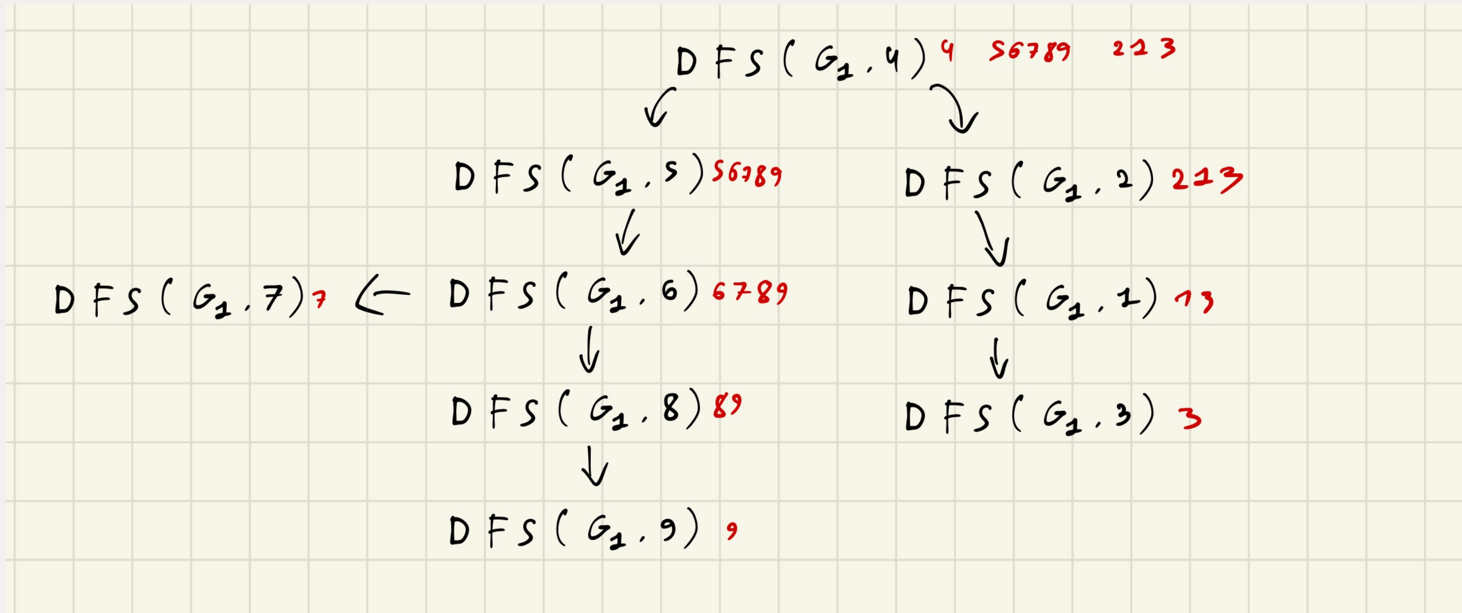
Question 3

On considère le graphe G_3 et le parcours en profondeur $(4, 5, 6, 7, 8, 9, 2, 1, 3)$. Comparer les intervalles $[pre(u), post(u)]$ et $[pre(v), post(v)]$ pour chaque arc avant (u, v), puis pour chaque arc arrière (u, v) puis pour chaque arc transverse (u, v).

Que remarque-t-on ? Pouvez vous l'expliquer ?

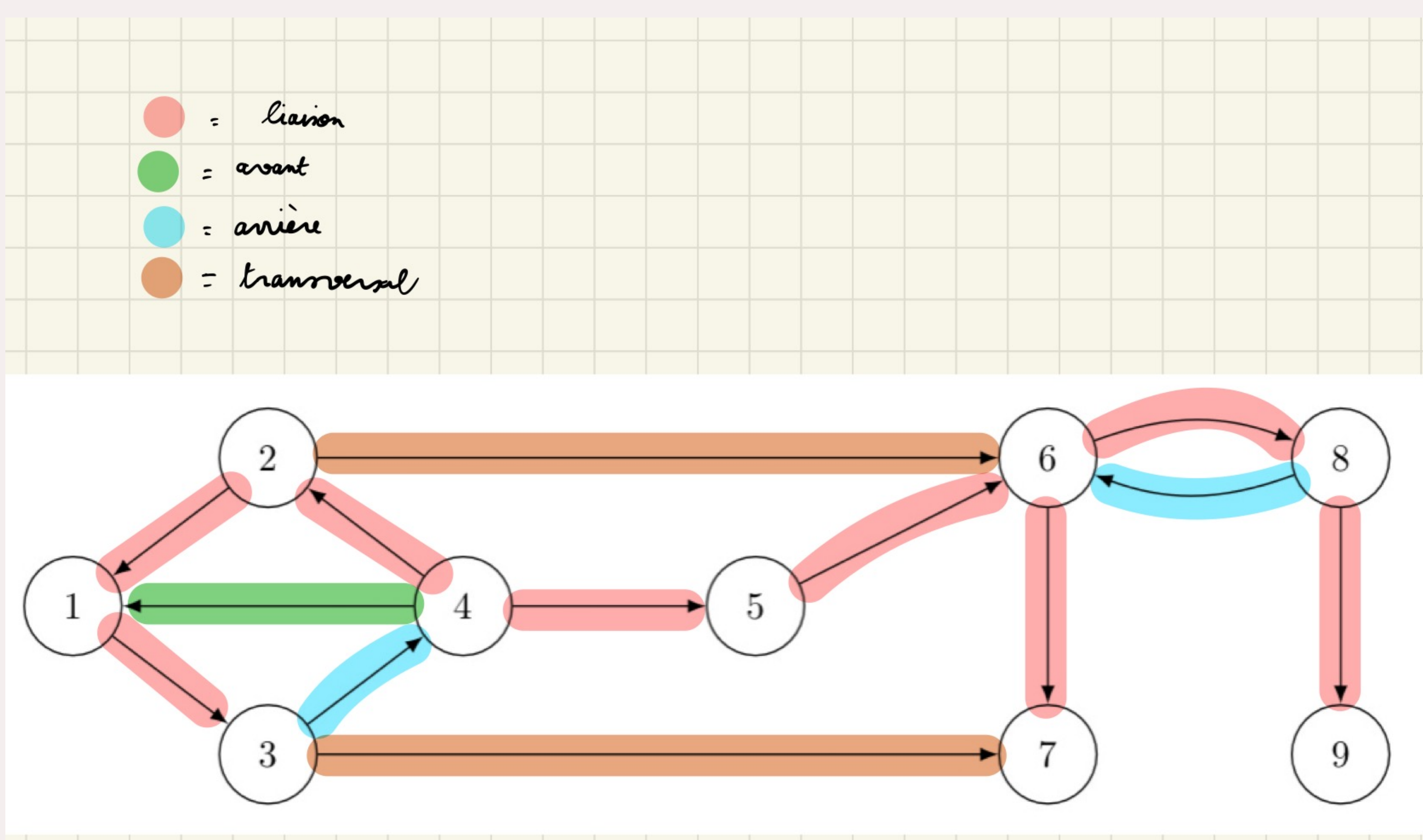
💡 **Réponses :**

▼ **Question 1 :**



▼ **Question 2 :**

- Classification des arcs :
  - Arc de liaison (arc du parcours) :
    - $\{(4,5),(5,6),(6,7),(6,8),(8,9),(4,2),(2,1),(1,3)\}$
  - Arc avant ($(u,v) \notin$ parcours, u apparait avant v dans le parcours, v est lié à un descendants)
    - $\{(4,1)\}$
  - Arc arrière ((u,v) avec u qui est un descendant de v)
    - $\{(8,6),(3,4)\}$
  - Arc transversal (tous les autres.)
    - $\{(3,7),(2,6)\}$, 3 et 7 ne sont descendant l'un de l'autre ! (Voir dans l'arbre)



▼ **Question 3 :**

- Si (u,v) est un arc de liaison : $[pre(v), post(v)] \subset [pre(u), post(u)]$
- Si (u,v) est un arc avant : De même.
- Si (u,v) est un arc arrière : $[pre(u), post(u)] \subset [pre(v), post(v)]$
- Si (u,v) est transversel : $[pre(v), post(v)] \wedge [pre(u), post(u)]$

u	pre(u)	post(u)
4	1	18
5	2	11
6	3	10
7	4	5
8	6	9
9	7	8
2	12	17
1	13	16
3	14	15

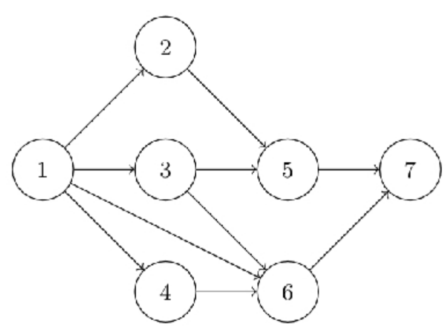
Exercice 1 :

Enoncé :

En préambule de cet exercice, vous pouvez regarder la courte vidéo de Christian Laforest :« Le parcours en profondeur (DFS) pour planifier (= tri topologique d'un graphe orienté sans circuit) »qui explique comment on peut utiliser un parcours en profondeur pour construire un ordre topologique d'un graphe orienté sans circuit.

Question 1

Dans cette question, on considère le graphe orienté sans circuit $G = (V, A)$ représenté par la figure suivante :



1. Exécutez un parcours en profondeur de ce graphe d'origine 1. Vous prendrez en priorité le sommet de valeur minimale quand plusieurs choix sont possibles. Vous préciserez l'arbre des appels, l'ensemble des sommets visités appel par appel, le graphe de liaison et la décomposition des arcs, et les intervalles $[pre(u), post(u)], u \in V$.
2. Est-ce que le graphe possède des arcs arrières pour L ? Pourquoi ?
3. Ordonnez les sommets selon les valeurs $post$ décroissantes. Qu'observez vous ?

Question 2

On suppose dans cette question que $G = (V, A)$ est un graphe orienté quelconque et que L est un parcours en profondeur. Soit alors $(u, v) \in A$ un arc quelconque de G.

— Comparer $post(u)$ et $post(v)$ en fonction du type d'arcs (liaison, avant, arrière, et transverse).

— En déduire que $post(u) < post(v)$ si est seulement si (u, v) est un arc arrière.

Question 3

On suppose maintenant que $G = (V, A)$ est un graphe orienté sans circuit et que L est un parcours en profondeur.

1. Montrez que pour tout arc $e = (u, v) \in A$, $post(u) > post(v)$;
2. Soit alors la liste $p = (v_1, \ldots, v_n)$ composée des éléments de V dans l'ordre des valeurs $post(u)$ décroissante. Montrez que p est un tri topologique de G.

Question 4

En déduire en quelques phrases un algorithme de calcul d'un tri topologique qui utilise un parcours en profondeur.

💡 **Réponses : Fallait être en cours j'ai eu une flemme INTENSE.**

- **Question 1 (2,3) :**
  - Il existe un arc arrière $\Leftrightarrow$ il existe un circuit.
  - Or, aucun circuit dans $G \Rightarrow$ pas d'arc arrière.
  - Ordonner les sommets selon la valeur post décroissante est un parcours (tester le sur l'exo 6 !)
- **Question 2 (voir exo6) :**
  - (u, v) arc arrière $\Leftrightarrow (post(u) < post(v))$
- **Question 3 :**
  - Pas de circuit $\Rightarrow$ pas d'arc arrière

    $$\Rightarrow (post(u) > post(v))$$
  - $(u, v) \in A$ avec $(post(u) > post(v))$

    $\Rightarrow u$ apparaît avant v dans p

    $\Rightarrow p$ est un tri.
- **Question 4 :**
  - Algorithme en entrée un graph orienté G (potentiellement sans racine) et en sortie un tri.
    - Cas sans racine :
      - Soit sur nouveau sommet que l'on relie à tous les sommets $u \in V$
      - Donc s est une racine. (On a crée un chemin de s vers tous les autres sommets. On nomme ce graphe G')
      - On appelle `DFS(G', s)` et on resort la liste des post ordonné par ordre décroissant. Par $Q.2$, c'est un tri.