TD2

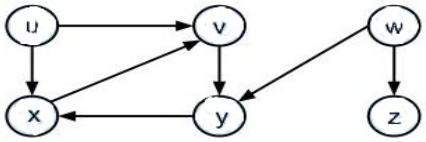
Exercice1

1/

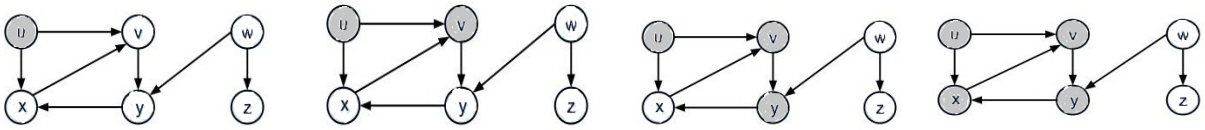
L'algo DFS associe à tout sommet $u \in X$:

- un temps de coloriage en gris d[u] : premier passage
  et
- un temps de coloriage en noir f[u] : dernier passage

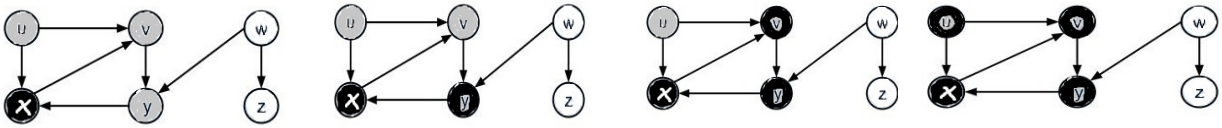
On colorie en blanc tout sommet $u \in X$



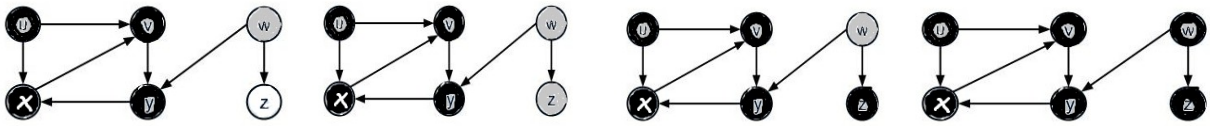
On colorie en gris u puis en gris tous les descendants sucessifs.



Quand il n'y a plus de sommet blanc a parcourir dans la liste des successeurs on remonte en coloriant en noir.



On recommence sur les arbres qui sont toujours en blanc.



On obtient ainsi avec l'algo en partant de l'arborecence de u:

d[u]=0 ; d[x]=1 ; d[y]=2 ; d[x]=3 ; f[x]=4 ; f[y]=5 ; f[v]=6 ; f[u]=7 ;
d[w]=8 ; d[z]=9 ; f[z]=10 ; f[w]=11 ;

2/

Dans un graphe G = (X;A) (orienté ou non) pour deux sommets quelconque u et v (fixés), une et une seule des conditions est vérifiée :

- Les intervalles [d[u]; f[u]] et [d[v]; f[v]] sont complètement disjoints

- L'intervalle [d[u]; f[u]] est entièrement inclus dans l'intervalle [d[v]; f[v]] et u est descendant de v dans l'arborescence en profondeur correspondante.

- L'intervalle [d[v]; f[v]] est entièrement inclus dans [d[u]; f[u]] et v est un descendant de u dans l'arborescence en profondeur correspondante.

Cela correspond aux trois cas possible pour deux sommets quelconque:
1) a est dans l'arborescence de b (dans l'algo)
2) b est dans l'arborescence de a (dans l'algo)
3) a et b sont 2 arborescences (dans l'algo)

cas 1) : DFS assigne (d[b],d[a],f[a],f[b]), ces variables dépendent de "temps" or la variable temps est strictement croissante.
On a alors $(d[b], f[b]) \supset (d[a], f[a])$.

cas 2) : DFS assigne (d[a],d[b],f[b],f[a]), ces variables dépendent de "temps" or la variable temps est strictement croissante.
On a alors $(d[b], f[b]) \subset (d[a], f[a])$.

cas 3) : Admettons a est colorié en noir avant que b soit en gris alors DFS assigne (d[a],f[a],d[b],f[b]).
On a donc les inervalles (d[a],f[a]) et (d[b],f[b]) disjoint.

3/
Le sommet v est un descendant propre du sommet u dans la forêt en profondeur d'un graphe G ⇔ d[u] < d[v] < f[v] < f[u].

⇒) Si v descend de u par l'algo on est dans le cas 2) et donc d[u] < d[v] < f[v] < f[u] car la variable temps est strictement croissante et les d[..] et f[..] dependent de "temps".

⇐) Si d[u] < d[v] , on a rencontré u avant v
de plus avec f[v]<f[u] , on a fini de traiter les descendants de u aprés v, donc v est dans l'arborescence de u.

4/
L'initialisation demande un temps en O(|X|).
La procédure Visiter_DFS est appelée exactement une fois sur chaque sommet.
La complexité des boucles "Pour chaque..." de tous les appels Visiter_DFS.
On a donc un algorithme linéaire en O(|S|+|A|).

Exercice 2

1/
Soit G = (X;A) un graphe orienté acyclique et la relation binaire $<_G$ sur X dénie par :
$u <_G v$ ⇔ ∃ un chemin de u à v dans G.

- $\forall u \in X$, le chemin nul mène au sommet lui-même ⇒ $u <_G u$
  Donc $<_G$ est reflexive.
- On est dans un graphe acyclique donc il n'existe pas un chemin retour (chemin+chemin de retour = cycle)
  Donc $<_G$ est anti symétrique.
- $\forall u, v, w \in X$ Tels que $u <_G v$ et $v <_G w$ alors ∃ un chemin de u vers v puis v vers w
  Alors ∃ un chemin de u vers w (en passant par v) ⇒ $u <_G w$
  Donc $<_G$ est transitive
- Si u et v sont deux puits alors il n'existe pas de chemin de u à v ni de v à u ⇒ $u !<_G v$ et $v !<_G u$
  Donc $<_G$ est partielle

Donc $<_G$ est une relation d'ordre partiel sur X.