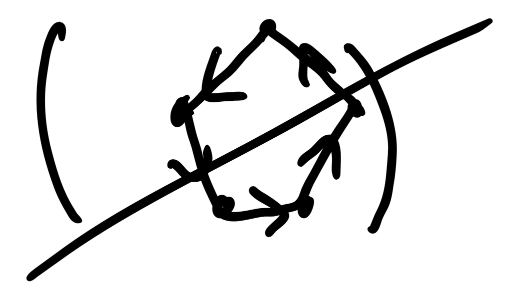
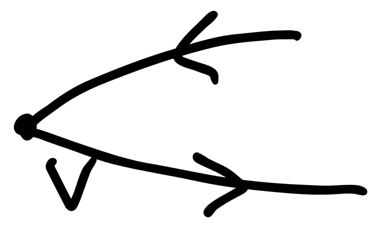
Def : un tri topologique d'un graphe orienté $G = (V, E)$ est un ordre total $<$ sur V t.q. pour tout arc $(u,v) \in E$, $u < v$ (càd, les sommets peuvent être placés sur une droite, et tous les arcs pointent dans la même direction.)

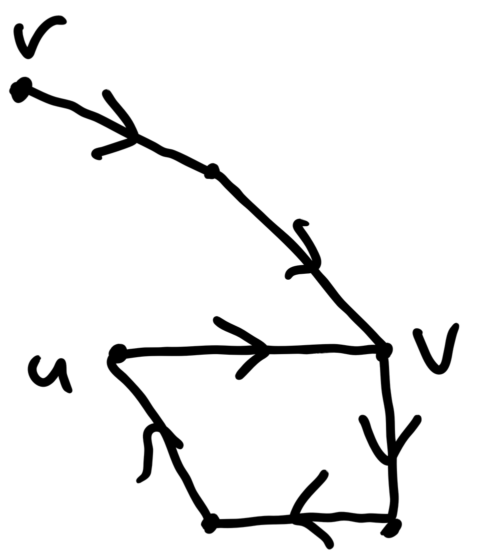
Théorème : Un graphe orienté G admet un tri topologique ssi G est acyclique



Preuve : Si G contient un cycle orienté C, alors C (et donc G aussi) n'admet pas un tri topologique.



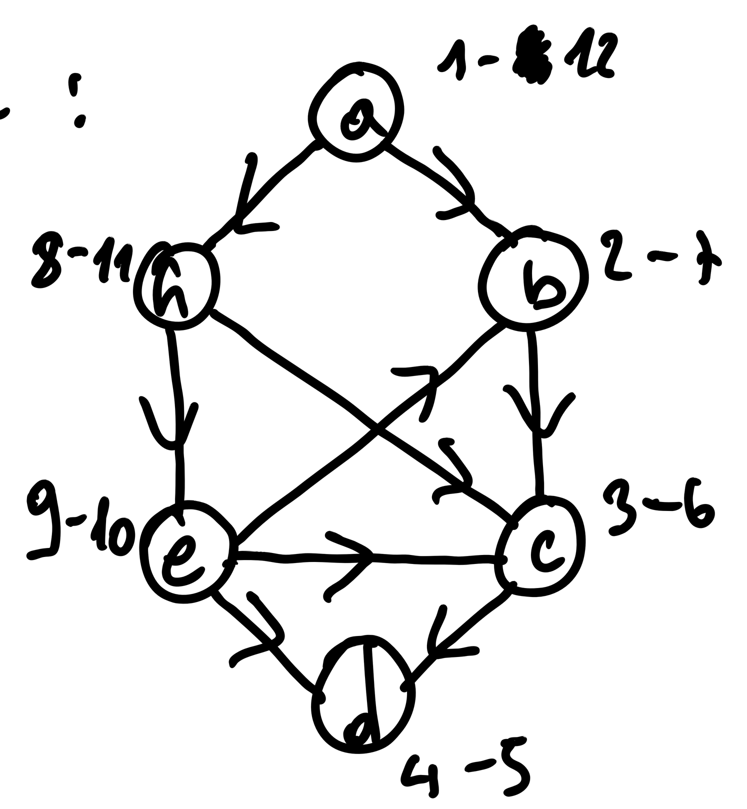
Inversement, supposons que G est acyclique. On définit un ordre total $<$ sur V comme suit : $u < v$ ssi $post(u) > post(v)$.
Soit (u,v) un arc quelconque de G. Comme G est acyclique, v n'est pas ancêtre de u.

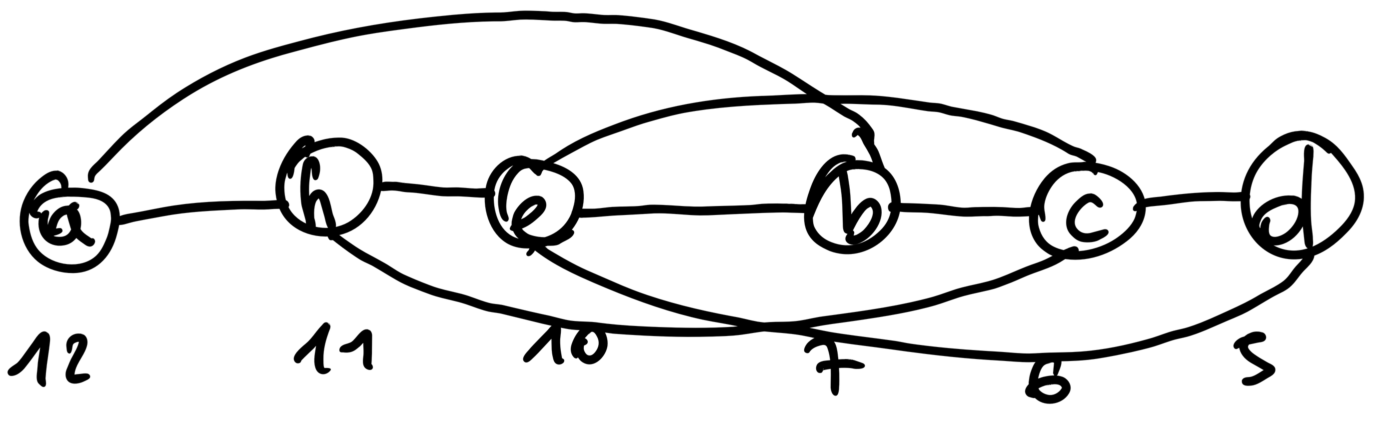


Donc, (u,v) n'est pas un arc "retour"
Donc, $post(u) > post(v)$
Donc, $u < v$

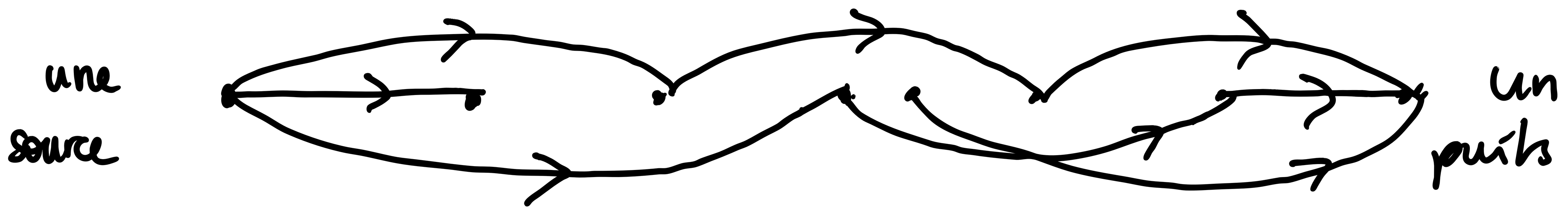
Remarque : En effectuant un DFS avec les tableaux pre et post, on peut ensuite trier les sommets par nombre post décroissant.

Exemple :





Remarques : Le plus petit sommet dans un ordre topologique est une "source" (aucun arc n'entre dans ce sommet)



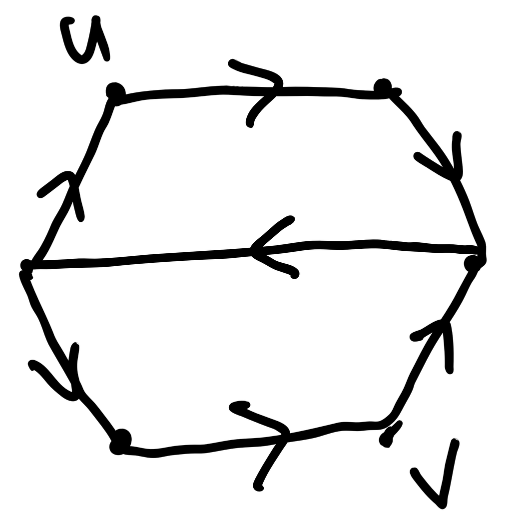
De même, le plus grand sommet dans cet ordre est un "puits" (aucun arc ne sort de ce sommet)

Théorème : Tout graphe orienté acyclique contient au moins une source et au moins un puits.

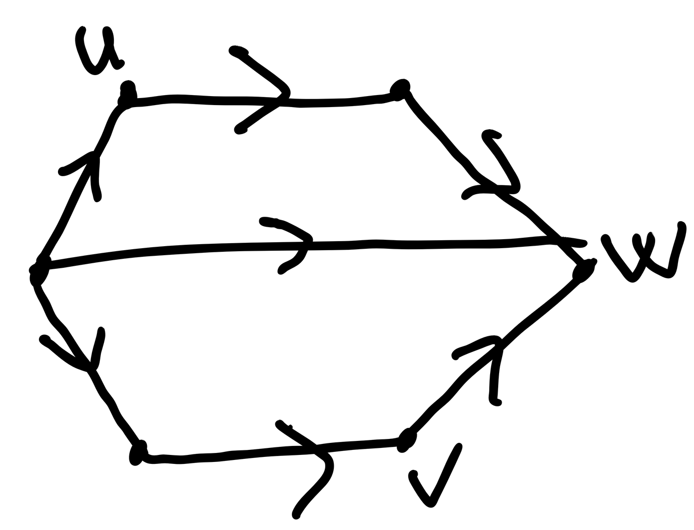
Ce théorème nous permet de trouver un tri topologique de la façon suivante : - Trouver une source et supprimer-la du graphe
- Répéter jusqu'à ce que le graphe devienne vide.

Connexité dans les graphes orientés
- Plus compliqué à définir dans le cadre des graphes orientés que c'était le cas pour les graphes non orientés.
- On dit que u et v sont connectés ssi ∃ un chemin de u vers v (respectant les flèches) et aussi il existe un chemin de v vers u.

Exemple :



∃ un chemin { v -> u ; u -> v



∄ chemin { u -> v ; v -> u

un chemin { u -> v ; w ↛ u

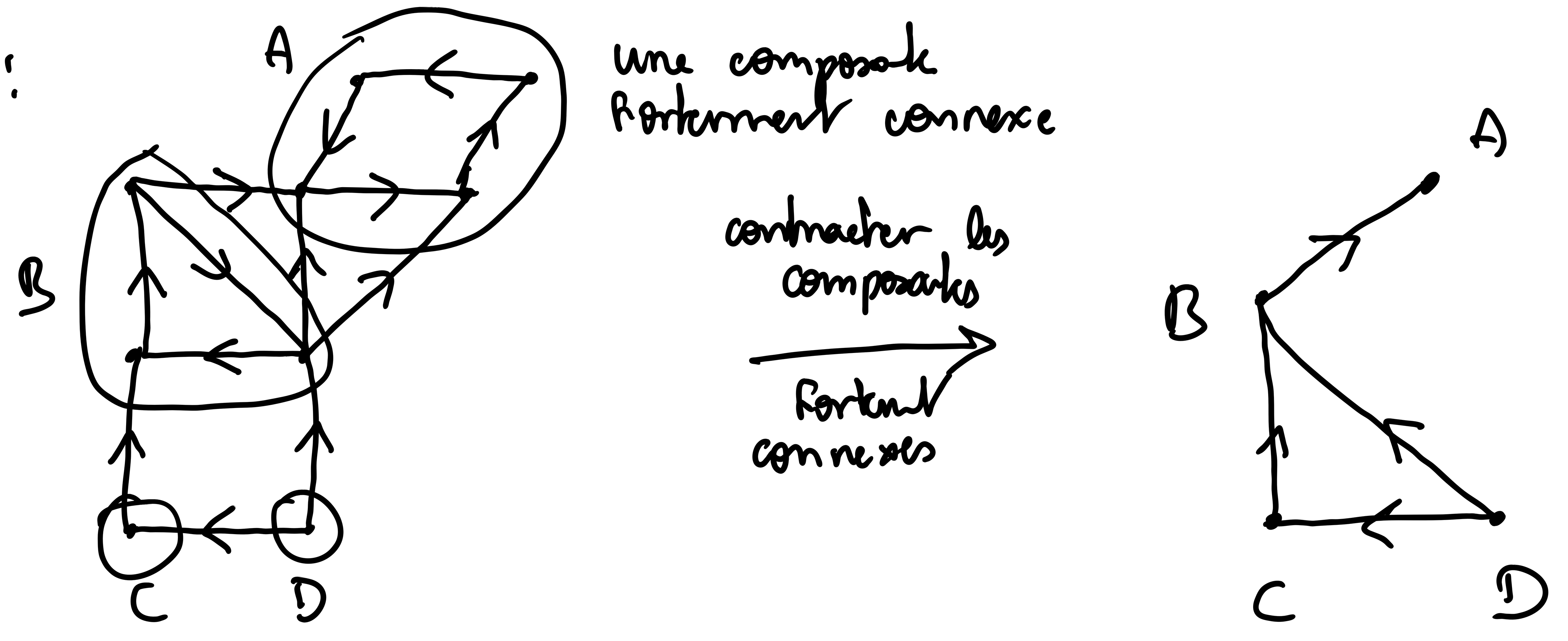
On va définir les composantes "fortement connexes" dans les graphes orientés.
Soit ~ la relation définie par :

u ~ v ssi u et v sont connectés (comme exemple ci-dessus)

On peut vérifier que ~ est une relation d'équivalence
- symétrique (u ~ v alors u ~ v)
- réflexive (u ~ v)
- transitive (u ~ v et v ~ w alors u ~ w)

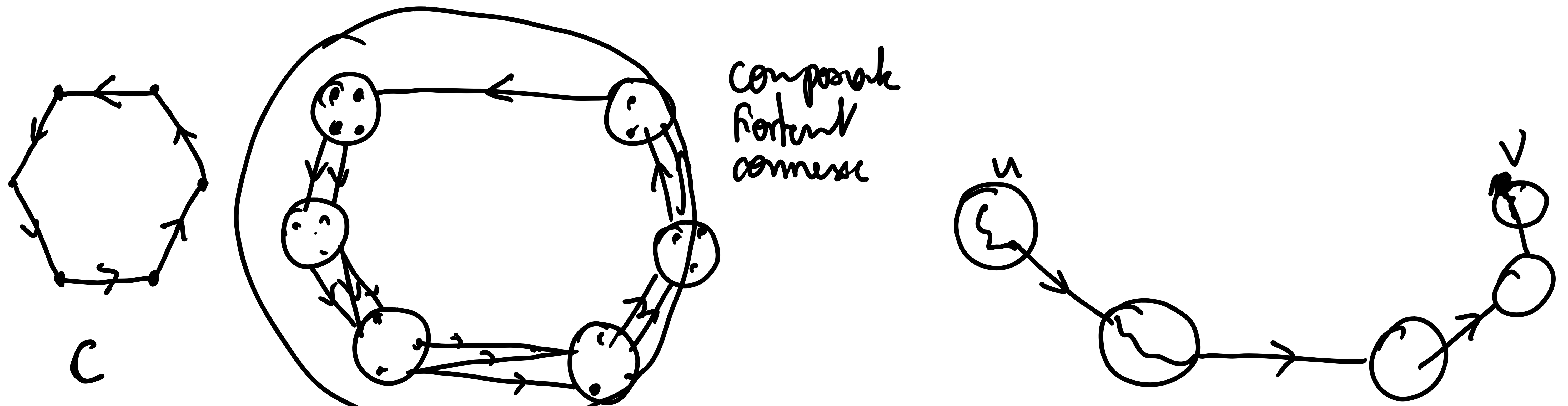
Les classes d'équivalence définies par ~ sont les composantes fortement connexes.
Une autre façon de voir les composantes fortement connexes : ce sont des sous graphes induits maximaux où chaque paire de sommets est connectée.

Exemple :



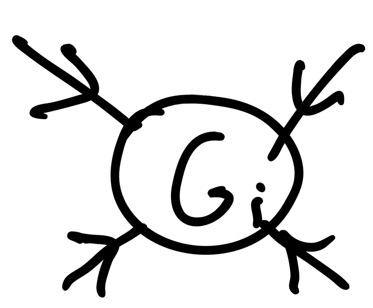
Théorème : Soit G un graphe orienté quelconque. Soit G' le graphe qu'on obtient en contractant les composantes fortement connexes de G. Alors, G' est acyclique.

Preuve : Supposons par l'absurde qu'il existe un cycle orienté C dans G'.



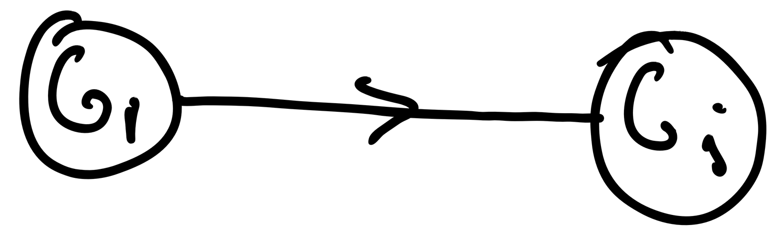
Propriété 1
Si la procédure explorer est lancée au sommet u, elle terminera précisément lorsque tous les sommets atteignables à partir de u auront été visités.

Donc, si u est un sommet d'une composante fortement connexe G_i de G qui est un puits dans le graphe contracté G', alors explorer (u) va parcourir précisément les sommets de G i et rien de plus



Propriété 2: Soient G_i et G_j des composantes fortement connexes de G. S'il existe un arc d'un sommet de G_i vers un sommet de G_j, alors ~~max~~ $\max\{post(u) : v \in V(G_i)\} \geq \max\{post(v) . v \in V(G_j)\}$

Preuve : Deux cas à considérer.

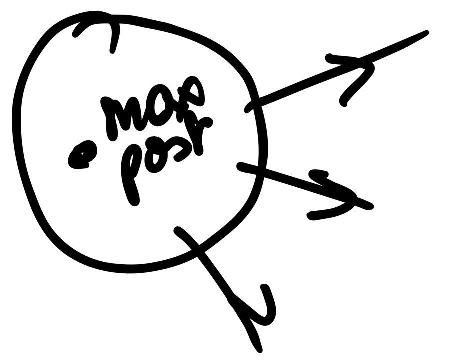


① supposons que DFS visite G_i avant G_j
Le parcours DFS va visiter tous les sommets de G_i et de G_j avant d'arrêter.

Le nombre $post(v)$, où v est le premier sommet de G_i visité ser superieur à $post(u)$, pour tout sommet dans $(V(G_i) \cup V(G_j)) \setminus \{v\}$

② Si DFS visite G_j en premier, alors la procedure explorer va s'arrêter après avoir visité tous les sommets de G_j, mais avant d'avoir visité tous les sommets de G_i.

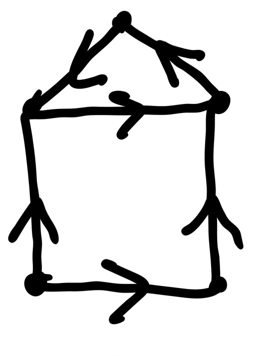
Propriété 3 : Le sommet avec la valeur maximale de post dans un parcours en profondeur appartient à une composante fortement connexe de type "source".



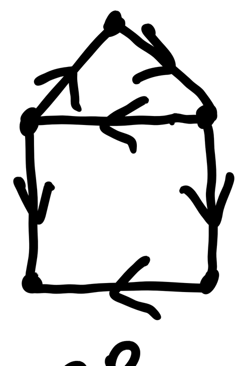
Conséquence : On peut trier les composantes fortement connexes par ordre décroissant de leurs nombres post maximaux.

Pourtant, on voudrait plutôt trouver un sommet dans une composante F.C. (fortement connexe) de type "puits"
Il suffit de considérer le graph inverse $G^R = (V, E^R)$, où $(u,v) \in E \Leftrightarrow (v,u) \in E^R$

Exemple :



G

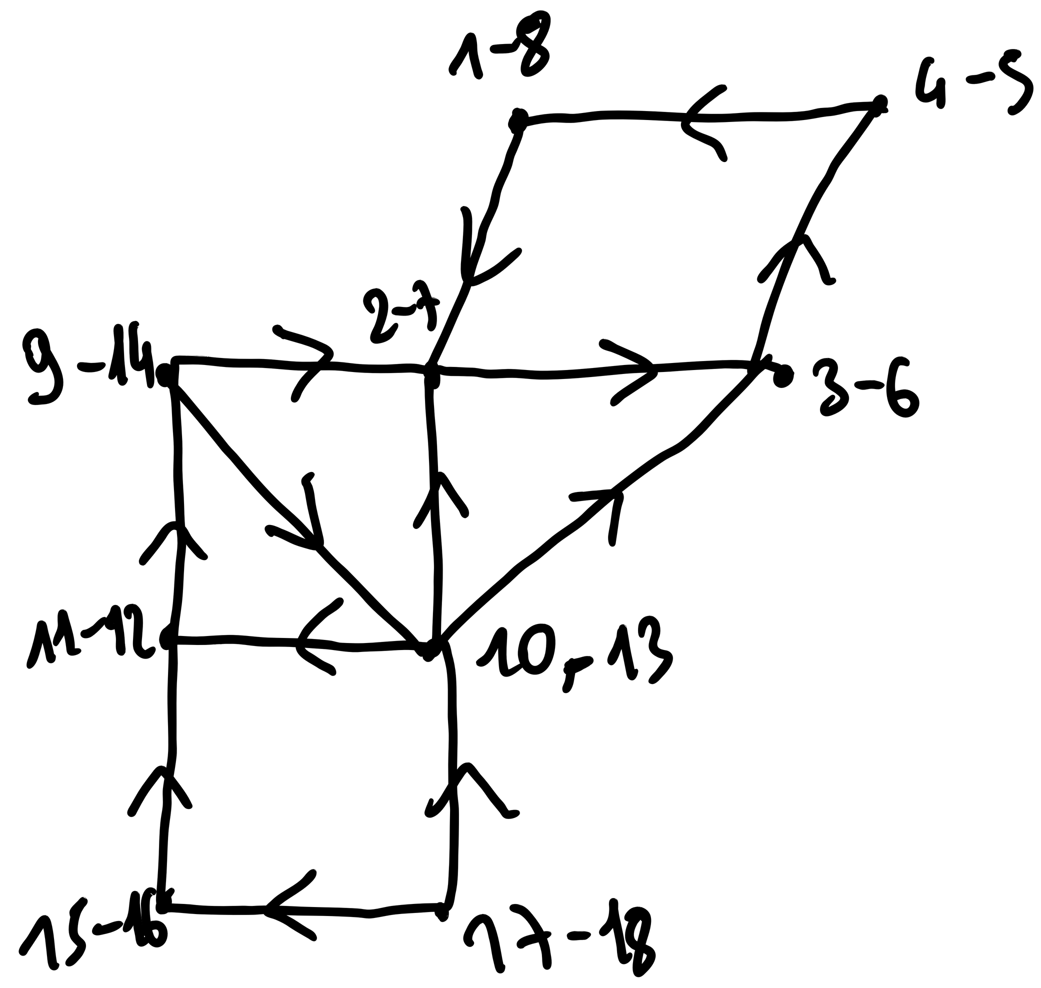


G^R

Notez que les composantes F.C. de G sont les mêmes que celles de G^R!

Algorithme pour calculer les composantes fortement connexes :

1. Exécuter DFS sur G^R
2. Exécuter DFS sur G, dans l'ordre de nombre post (trouvé dans l'étape 1) décroissant.



prochain cours...