AMD5
TD n° 4 : Tri topologique, Composantes fortement connexes

I) Tri topologique

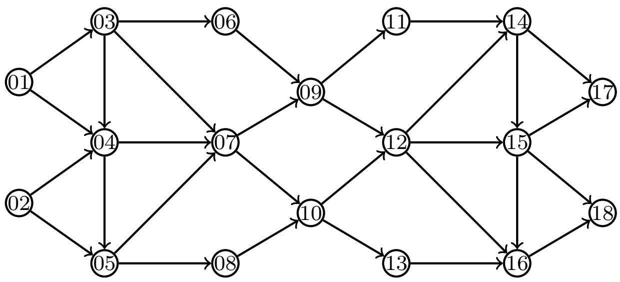
Rappel de la définition : Soit $G = (S, A)$ un graphe orienté ; un *tri topologique* de G est une énumération $(s_1, s_2, \dots, s_n)$ des éléments de S telle que, pour tous i, j, on ait $(s_i, s_j) \in A \Rightarrow i < j$.

Exercice 1 : Calcul par parcours en profondeur

1. Rappeler l'algorithme de tri topologique d'un graphe orienté acyclique (Directed Acyclic Graph, ou DAG en anglais) basé sur un parcours en profondeur vu en cours. Quelle est sa complexité ?

2. Modifier l'algorithme pour qu'il détecte si le graphe n'admet aucun tri topologique (autrement dit s'il n'est pas un DAG)

3. Appliquer l'algorithme sur le graphe suivant.



Exercice 2 : Tri topologique - autre algorithme

1. Montrer que si un graphe admet un tri topologique alors il ne contient aucun circuit orienté.

2. Montrer la réciproque en montrant qu'un DAG possède nécessairement une source (un sommet de degré entrant 0). Est-il aussi vrai qu'un tel graphe possède nécessairement un puits (degré sortant 0) ?

3. En se basant sur cette observation proposer un algorithme permettant de construire le tri topologique d'un DAG. Quelle est sa complexité ?

4. Améliorer la complexité en construisant, puis en maintenant une liste de tous les sommets qui peuvent être le prochain dans l'ordre. Quelle est la nouvelle complexité ?

II) Composantes fortement connexes

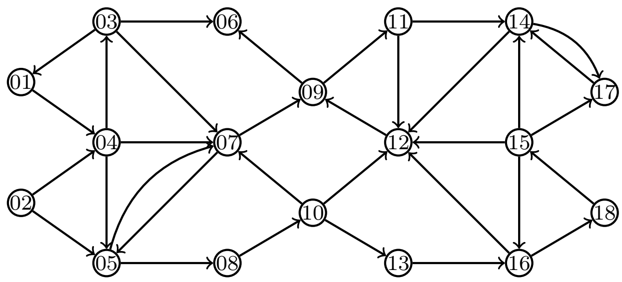
Exercice 3 : Algorithme de calcul

On rappelle l'algorithme vu en cours :

1. Appeler `PPC`(G) pour obtenir les dates `f` de chaque sommet ;
2. Construire G^t (le graphe transposé de G) ;
3. Appeler `PPC`(G^t) en énumérant les sommets par date `f[-]` décroissante ;

4. Retourner les arbres de parcours obtenus par `PPC`(G^t) : ce sont les CFC.

Application. Appliquer l'algorithme de composantes fortement connexes sur le graphe suivant.



Implémentation. Comment implémenter les étapes 2 et 3 pour avoir un algorithme en temps linéaire ?

Nombre de composantes. Écrire tout le pseudo-code d'un algorithme qui compte le nombre de composantes fortement connexes d'un graphe.

Graphe acyclique. Montrer que si le graphe G est un DAG, chaque relance à l'étape 3 ne visite qu'un seul sommet. Dans quel ordre ces relances sont-elles faites ?

Moins de profondeur. Peut-on remplacer l'étape 1 par un parcours en largeur ? Et l'étape 3 ?

Exercice 4 : Chemins et cycles hamiltoniens

Definitions :

- On appelle *chemin hamiltonien* sur un graphe un chemin passant par chaque sommet du graphe une fois et une seule.
- Un *cycle hamiltonien* est un cycle passant par chaque sommet du graphe une fois et une seule.
- Un *tournoi* est un graphe orienté tel que pour tout paire de sommets u,v soit $(u, v) \in A$ soit $(v, u) \in A$.

1. Si un graphe possède un cycle hamiltonien, que dire de son nombre de composantes connexes ?

2. Réciproquement, si un graphe est fortement connexe, est-il nécessaire qu'il ait un cycle hamiltonien ?

3. On a vu exercice 2 qu'un graphe orienté acyclique (DAG, pour *Directed Acyclic Graph*) admet un tri topologique. Montrer qu'un DAG n'a qu'un seul tri si et seulement s'il possède un chemin hamiltonien.

4. Montrer qu'un tournoi G possède un chemin hamiltonien.

5. Montrer qu'un tournoi fortement connexe possède un cycle hamiltonien.