Université Polytechnique Hauts-de-France
Institut National des Sciences Appliquées Hauts-de-France **2020-2021**

Spécialité Informatique par Apprentissage 1ère année Graphes et Algorithmes

Fiche d'exercices pratiques 5

Vous utiliserez une des deux structures de données pour représenter les graphes, au choix.

Exercice 1 :

On considère des graphe **non orientés**.

1. Implémenter l'algorithme de `Warshall` pour calculer la fermeture transitive du graphe et en déduire les composantes connexes du graphe. Tester cette méthode sur des graphes de petite taille ($|S| \leq 20$).

On s'intéresse maintenant à des graphes **orientés** et à la recherche des composantes fortement connexes de ces graphes. On considère l'algorithme de `Kosaraju`. Soit G le graphe initial, le principe est le suivant :

(i) Appliquer un parcours en profondeur pour calculer les dates de fin de traitement de chaque sommet.

(ii) Calculer le graphe transposée de G, graphe dans lequel on inverse le sens de chaque arc.

(iii) Appliquer le parcours en profondeur sur le graphe transposée mais en considérant les sommets dans l'ordre décroissant des dates de fin de traitement du premier parcours.

(iv) Chaque arbre généré par l'étape précédente est une composante fortement connexe de G.

1. Implémenter cette approche.
2. Ajouter un algorithme permettant de générer le graphe réduit.

Exercice 2 :

Implémenter l'algorithme de `Kahn` (voir algorithme 1) pour faire un tri topologique d'un graphe **orienté sans circuit** donné. Dans cet algorithme le tableau `n` contiendra le niveau de chaque sommet.

Algorithme 1 : Algorithme de Khan

```
F ← ∅ ; // F file
pour x ∈ S faire
    n[x] ← +∞ ; // n[x] est le niveau du sommet x
    d⁻[x] ← degré entrant de x ;
    si d⁻[x] = 0 alors
        F ← F ∪ x ;
        n[x] ← 0 ;
    finsi
finpour
c ← 0 ;
tant que F ≠ ∅ faire
    x ← tête(F) ; Défiler(F) ;
    c ← c + 1 ;
    for chaque y successeur de x do
        d⁻[y] ← d⁻[y] - 1 ;
        si d⁻[y] = 0 alors
            Enfiler(F, y) ;
            n[y] ← n[x] + 1 ;
        finsi
    end
fintq
si c ≠ n alors
    écrire "tri topologique impossible" ;
finsi
```
