Contrôle Continu INFO1 - P3
Ressource R2.07

Nom Responsable	Godin Thibault
Date contrôle	29/03/2022
Durée contrôle	1h30 min
Nombre total de pages	3
Impression	recto/verso
Documents autorisés	A4 recto-verso
Calculatrice autorisée	NON
Réponses	3 copies

Les réponses doivent être justifiées et les raisonnements, calculs et théorèmes doivent apparaître clairement. La qualité de la rédaction et les efforts de recherche seront pris en compte

Copie 1

Exercice 1 : Éco-gestion

On souhaite transformer des salles de cours en salles info. La gestion des tâches est donnée par le tableau ci-dessous.

Tâche à réaliser	**Repère**	**Durée (en jours)**	**Tâches précédentes**
Vider la salle et démonter le matériel	A	2	–
Nettoyer et repeindre	B	4	A
installer table et tableau	C	1	B
Commander et réceptionner le câblage	D	10	–
Déballer et contrôler le câblage	E	1	D
Câbler la salle	F	3	B, E
Installer et brancher les postes info.	G	1	C, F
Installer les logiciel, configurer et tester les postes	H	7	G

1. Donner la matrice d'adjacence du graphe modélisant ce problème.
2. On suppose que l'IUT dispose de personnels qualifiés en quantité illimité pour travailler à la rénovation de ces salles.

a. Comment organiser le travail au mieux et dans quel ordre. Préciser l'algorithme utilisé et dessiner le graphe que vous trouvez en résultat (attention à bien faire apparaître chaque étape de l'algorithme sur votre copie).

b. Donner, pour chaque tâche, la date au plus tôt à laquelle elle peut commencer. En déduire la durée totale des travaux.

3. Par mesure d'économie, l'IUT décide de n'affecter qu'une seule personne à la rénovation.

   a. Donner un algorithme permettant de déterminer un ordre approprié.

   b. Proposer un ordre valide pour l'exécution de ces tâches.

   c. Donner la durée totale que prendront les travaux dans ce cas.

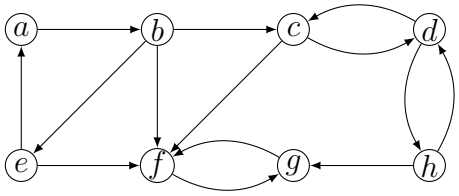
. . . . . .

Copie 2

. . . . . .

Exercice 2 : Composantes fortement connexes

L'algorithme de Kosaraju (1978) est un algorithme permettant de déterminer les composantes *fortement connexes*.



a. `Calculer un ordre postfixe du graphe`

b. `Inverser les arcs du graphe et faire un parcours dans l'ordre post-fixe renversé. À chaque fois que la pile est vide on obtient une nouvelle composante fortement connexe.`

1. Soit G_1 le graphe dessiné. Donner sa matrice d'adjacence, puis dessiner le graphe $\overleftarrow{G_1}$ où l'on a inversé le sens des arcs et donner sa matrice d'adjacence.

2. Écrire en pseudo-code un algorithme prenant en entrée la matrice d'adjacence A d'un graphe orienté G et qui renvoie la matrice d'adjacence $\overleftarrow{A}$ du graphe où l'on a inversé le sens des arcs.

3. Appliquer l'algorithme de Kosajaru à G_1, et donner ses composantes fortement connexes.

4. Donner un algorithme permettant de trouver les composantes connexes dans un graphe non-orienté.

Exercice 3 : La SNCF doit transporter des produits chimiques en train. Malheureusement ce transport est potentiellement dangereux et certains produits ne doivent en aucun cas être en contact avec d'autres. Le train doit donc être séparé en wagons transportant chacun des produits chimiques n'interagissant pas ensemble.

La Brochure ED 753 *Stockage et transfert des produits chimiques dangereux* résume les incompatibilités dans le tableau suivant.

	A	B	C	D	E	F	G	H	I	J
A		X			X				X	
B	X							X		X
C							X	X	X	
D						X			X	X
E	X					X	X			
F				X	X			X		
G			X		X					X
H		X	X			X				
I	X		X	X						
J		X		X			X			

1. Donner une première estimation du nombre de wagons nécessaires avec l'algorithme de coloration naïve (en précisant bien toutes les étapes et l'ordre choisi).
2. On rappelle l'algorithme DSatur : **DSatur Algorithm (so called because it uses saturation degree)**
   (a) `Arrange the vertices by decreasing order of degrees.`
   (b) `Choose a vertex with a maximal saturation degree. If there is an equality, choose any vertex of maximal degree and saturation in the uncolored subgraph.`
   (c) `Color the chosen vertex with the least possible (lowest numbered) color.`
   (d) `If all the vertices are colored, stop. Otherwise, return to` (b)

   Appliquer *en précisant bien toutes les étapes* l'algorithme DSatur à ce graphe.
3. Donner inférieure sur le nombre optimal de wagons utilisés (bien préciser comment vous obtenez cette borne).
4. Écrire (en pseudo-code ou avec une syntaxe proche du python), une fonction `VerifColoration` ; qui prend en paramètre une matrice d'adjacence `A` et un tableau `C` donnant la couleur de chaque sommet ; et qui renvoie `Vrai` si `C` correspond à une coloration valide du graphe représenté par `A`, `Faux` sinon.