Algorithmique procédurale avancée
Examen - ING1 GI

Lundi 13 mai 2019

Modalités

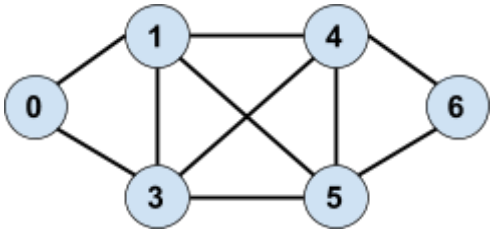
- Durée : 2 heures
- Vous devez rédiger votre copie à l'aide d'un **stylo à encre** exclusivement.
- Toutes vos affaires (sacs, vestes, trousse, etc.) doivent être placées à l'avant de la salle.
- Aucun document n'est autorisé.
- Aucune machine électronique ne doit se trouver sur vous ou à proximité, même éteinte.
- Aucun déplacement n'est autorisé.
- **Aucune question au professeur n'est autorisé**. Si vous pensez avoir détecté une erreur, continuez en expliquant les hypothèses que vous faites.
- Aucun échange, de quelque nature que ce soit, n'est possible.
- Le barème est donné à titre indicatif.

Exercice 1 : Cours et application directe (5 pts)

1. Soit un graphe (simple et sans boucle) non orienté et connexe. Quel est son nombre minimum d'arêtes ? Quel est son nombre maximal d'arêtes ?
2. Quel est le coût du calcul de l'ensemble des successeurs d'un sommet dans un graphe orienté représenté par une matrice d'adjacence ? une liste d'adjacence ?
3. Quel est le coût du calcul de l'ensemble des prédécesseurs d'un sommet dans un graphe orienté représenté par une matrice d'adjacence ? une liste d'adjacence ?
4. Comment définit-on la taille d'un graphe ?
5. Peut-on écrire un parcours en profondeur en récursif ? en itératif ?

Exercice 2 : Cycle eulérien (4 pts)

Trouver un cycle eulérien dans un graphe se fait en plusieurs étapes. Voici le déroulé d'un algorithme sur le graphe G suivant :



a. Construire un cycle **quelconque** C :
   C = 1-4-3-5-1
b. Si toutes les arêtes ne sont pas dans le cycle, alors
c. Déterminer les composantes connexes de G privé des arêtes de C :
   H_1 {0,1,3} et H_2 {4,6,5}
d. Pour chaque composante connexe H_k :
   i. Trouver un cycle **eulérien** E_k:
      E_1 = 3-0-1-3 (puis E_2 = 4-5-6-4)
   ii. Intercaler le cycle eulérien E_k dans le cycle quelconque C
      C = 1-4-3-0-1-3-5-1 (puis C = 1-4-5-6-4-3-0-1-3-5-1)
e. Le cycle C est eulérien pour le graphe G

1. Quelle(s) condition(s) doi(ven)t être satisfaite(s) pour qu'un graphe contienne un cycle eulérien ?
2. Ecrire une fonction qui renvoie un cycle (quelconque) d'un graphe.
3. Ecrire une fonction qui donne les composantes connexes d'un graphe.
4. Ecrire une fonction qui intercale un cycle dans un autre.
5. Ecrire une fonction qui permet de trouver un cycle eulérien.

Exercice 3 : coloration de graphes et algorithme DSAT (5 pts)

1. Donner la définition du nombre chromatique d'un graphe.
2. Quel algorithme vu en cours (qui trie les sommets par ordre décroissant de degré) permet de calculer une borne supérieure pour ce problème ?

   Dans la suite, on étudiera un algorithme qui utilise un ordre dynamique sur les degrés des sommets.

3. Soit DSAT(s) le nombre de couleurs différentes parmi les voisins d'un sommet s.
   On part d'un graphe dont aucun des sommets est colorié.
   A chaque itération de l'algorithme, on choisit le sommet dont le DSAT est maximum.
   En cas d'égalité, on choisit le sommet de degré le plus grand. En cas de nouvelle égalité, on choisit le sommet d'indice le plus petit (cf. règle de Bland en optimisation linéaire).
   Une fois le sommet choisi, on lui attribue la plus petite couleur possible.
   On répète ce procédé jusqu'à ce que tous les sommets soient coloriés.
4. Soit le graphe dont les 8 sommets sont les entiers {1,2,...,8} et dont les 12 arêtes sont {(1,2),(1,3),(1,4),(1,5),(2,3),(2,4),(2,6),(5,7),(5,8),(6,7),(6,8),(7,8)}.
   Quel est son nombre chromatique ?
   Appliquer l'algorithme sur ce graphe. Combien de couleurs l'algorithme utilise-t-il ?
5. Ecrire la fonction DSAT.

Exercice 4 : Implémentation naïve et implémentation efficace de l'algorithme de Kruskal (6 pts)

1. Que calcule l'algorithme de Kruskal ?
2. L'algorithme nécessite :
   a) de vérifier si deux sommets font partie de la même composante.
   b) de fusionner deux composantes si ce n'est pas le cas.
   Comment ces deux opérations sont-elles réalisées dans le cours ? Quelle est la complexité de chacune d'entre elles ? Quelle est alors la complexité totale de l'algorithme ?
3. On propose ici une méthode alternative pour accélérer l'union de deux composantes : on maintient un tableau d'entiers (dont la taille est le nombre de sommets du graphe). Ce tableau représente un ensemble d'arbres. Chaque arbre représente une composante connexe.
   Le fonctionnement du tableau est le suivant :
   - L'indice de chaque case du tableau est le numéro de sommet.
   - Le contenu de chaque case est :
     - soit l'indice de son père s'il en a un.
     - soit la taille de sa composante AFFECTEE D'UN SIGNE NEGATIF si le sommet est la racine d'un arbre.

   a. Ecrire une procédure **initialisation(cc : tableau de entier (S), n : Entier)** qui initialise les composantes sans considérer les arêtes.
   b. Ecrire un algorithme **accesComposante(s : Entier (E), cc : tableau de Entier (E)) : Entier** qui calcule le numéro de la composante d'un sommet. Quelle est sa complexité au pire ?
   c. Ecrire un algorithme **reunirComposantes(s1 : Entier, s2 : Entier, cc : tableau de Entier (E/S))** qui prend 2 sommets et qui réunit leurs composantes. On exploitera autant que possible la connaissance des tailles des composantes.
   d. Conclure sur les performances de cette méthode.